Contributions de l'inférence grammaticale à la fouille de données séquentielles

Stéphanie Jacquemont

Ecole doctorale de Saint Etienne

Contributions de l'inférence grammaticale à la fouille de données séquentielles

Thèse préparée
pour obtenir le grade de :

Docteur de l'université Jean Monnet de Saint-Étienne

Mention : Informatique.

par Stéphanie Jacquemont

Laboratoire Hubert Curien — Faculté des Sciences et Techniques

Soutenance le 4 décembre 2008, devant le jury composé de :

Jean-François Boulicaut ..... Professeur, INSA, Lyon, *Rapporteur*
Bertrand Braunschweig ..... Responsable de Programmes à l'ANR, Paris
Antoine Cornuéjols ..... Professeur, INA-PG, Paris
Bruno Crémilleux ..... Professeur, Université de Caen, Caen, *Rapporteur*
François Jacquenet ..... Professeur, UJM, Saint Etienne, *Co-Directeur*
Pascal Poncelet ..... Professeur, Ecole des mines d'Alès, Nîmes
Marc Sebban ..... Professeur, UJM, Saint Etienne, *Directeur*

Remerciements

Au terme de ces années de thèse qui me conduisent aujourd'hui à présenter ce manuscrit, c'est avec un réel plaisir que je profite de cette section pour remercier celles et ceux qui ont été associés, de prêt ou de loin, à ma vie de doctorante.

Je tiens à remercier Jean-François Boulicaut, Professeur à l'INSA de Lyon, d'avoir accepté d'être rapporteur de cette thèse. Jean-François Boulicaut est un spécialiste en fouille de données et je le remercie de s'être intéressé à mes travaux. C'est lui qui a eu le premier l'idée du lien original entre nos travaux initiaux et le domaine de la préservation de la vie privée. Je le remercie de nous avoir ouvert cet horizon passionnant.

Je souhaite remercier Bruno Crémilleux, Professeur à l'université de Caen, d'avoir bien voulu rapporter mon travail de thèse. Je tiens à le remercier pour la qualité des remarques faites sur mon mémoire. Son analyse sur notre travail est très intéressante.

Je désire également remercier Bertrand Braunschweig, Responsable de Programmes et futur directeur du département STIC à l'ANR, d'avoir accepté de participer à ce jury. Son intérêt pour les travaux développés par le consortium du projet Bingo a de toute évidence contribué au financement du projet Bingo2. Ancien président de l'AFIA, il a une vision affûtée des recherches en intelligence artificielle et c'est un honneur de le voir siéger dans ce jury.

Je remercie Antoine Cornuéjols, Professeur à AgroParisTech, de faire partie de mon jury de thèse. Je suis honorée d'avoir dans mon jury l'un des auteurs du livre français majeur sur l'apprentissage automatique.

Je souhaite remercier Pascal Poncelet, Professeur à l'école des Mines d'Alès, d'avoir accepté de faire partie de mon jury. Il est un des chercheurs reconnus dans le domaine de la fouille de données séquentielles et nombre de ses articles m'ont été très utiles.

Je remercie l'Agence Nationale de la Recherche pour avoir financé partiellement les travaux présentés dans ce manuscrit au travers des projets Bingo et Bingo2. Je remercie le réseau européen d'excellence PASCAL d'avoir également financé partiellement ces recherches, et de m'avoir permis de participer à l'école *Pattern Analysis* qui a été très fructueuse pour moi.

Je tiens à exprimer ma profonde gratitude envers mon directeur de thèse Marc Sebban, Professeur à l'université de Saint-Etienne. Je tiens à le remercier pour la qualité de son encadrement. Il m'a beaucoup aidé dans le déroulement de ce travail de thèse. J'ai apprécié sa manière de travailler, rigoureuse, constructive, présentant toujours les choses de façon très pédagogique. Il m'a transmis une image très positive de persévérance malgré les échecs. Grâce à sa grande rigueur et son expérience, il m'a permis de progresser sur bon nombre de points, que ce soit dans la démarche scientifique ou dans la communication écrite ou orale. Je le remercie de sa patience envers mes nombreuses lacunes. Je tiens également à le remercier d'être toujours resté disponible malgré ses nombreuses fonctions.

Je tiens également à remercier grandement mon co-directeur de thèse François Jacquenet, Professeur à l'université de Saint-Etienne. Il a lui aussi grandement participé à ma formation d'enseignant-chercheur. Je tiens tout particulièrement à le remercier pour son enthousiasme, mais aussi son réalisme. Son regard critique a aussi souvent été très bénéfique pour améliorer ce travail. De plus, en me faisant part de son expérience, il s'est souvent montré rassurant lorsque des épreuves angoissantes se sont présentées.

Tous deux m'ont donné le goût à la recherche, au travers des deux stages passionnants que j'ai pu effectuer avec eux au cours de mon master. Ils m'ont encouragé à continuer sur cette voie. Je tiens aussi à les remercier tous les deux d'avoir été compréhensifs envers mes choix personnels.

Je remercie également l'ancien directeur de l'EURISE, Colin De La Higuera, de m'avoir permis d'évoluer dans une structure accueillante, d'un point de vue matériel et humain. En considérant chaque doctorant comme un membre à part entière du laboratoire, il m'a permis de découvrir toutes les facettes du travail d'un enseignant-chercheur.

Un merci tout particulier à Thierry pour l'environnement technique de travail de qualité qu'il nous procure, ainsi que pour toutes ses aides techniques et ses conseils précieux.

Merci également à tous mes collègues informaticiens pour leur soutien Baptiste, Catherine, Christine, Christophe, Émilie, Fabrice, Franck, Frédéric, Jean, Jean-Christophe, Laurent, Marc B, Mathias, Phillipe, Pierre et sans oublier ceux qui sont partis : Amaury, Hazael, Henri-Maxime, Rémi, Sabri et Toufik. Une petite pensée particulière pour ceux avec qui je partage souvent des pauses café et des repas conviviaux.

Je remercie également tous mes amis de fac et d'ailleurs, Seb et Marie, Fred et Céline, Lise, François, Dav, Maryline et Mathieu, Pascale et Bruno, Pollux, Mimie pour leurs messages d'encouragements.

Merci à Colette et Roger de me permettre de venir travailler sereinement chaque jour.

Pour leurs encouragements et leur assistance aussi bien matérielle que morale tout au long de mes études je tiens à remercier mes parents, ma sœur ainsi que toute ma

famille. Un clin d'œil à mon papa qui, tant qu'il a pu, m'a toujours aidé dans mes études.

Enfin, merci à Sylvain pour son soutien quotidien, sa confiance en moi, ses petits coups de pouce dans les moments de doute. Surtout, je le remercie d'avoir enduré ces quatres années, parfois difficiles pour lui, pour que je mène à bien cette thèse. Ma dernière pensée va à ma petite Flora qui m'apporte tant de bonheur.

Sommaire

Introduction

Le volume d'information sauvegardé quotidiennement dans des bases de données ne cesse de croître chaque année. Ce phénomène est matérialisé au travers du VLDB survey [1] dont l'initiative permet d'identifier régulièrement les plus grandes bases de données de par le monde. Ainsi, entre 1997 et 2005, leur taille en situation commerciale est passée d'environ 3200 Go à 100000 Go, soit un accroissement d'un facteur 31 en à peine dix années. On constate que la barre des 100 To, qui aurait pu encore sembler mythique il y a peu, est donc d'ores et déjà franchie. Sur le plan de l'efficacité des SGBD en terme de nombre de transactions par seconde, là aussi la progression a été très importante. D'un record de 1820 transactions par seconde en 1997, nous sommes passés à plus de 8000 transactions par seconde (plus de 28 millions par heure) en 2005. L'étude VLDB montre également que tous les secteurs économiques sont touchés par cette inflation continuelle du volume d'informations collectées. On y trouve ainsi répertoriées des bases de données de Yahoo!, Amazon, UPS, France Télécom, Barclays Bank, Samsung, Kmart, etc. Ce phénomène est également visible sur le Web où la quantité d'informations accessibles ne cesse de croître très rapidement et où les internautes sont de plus en plus submergés par le volume d'information à leur disposition. Cette croissance a évidemment conduit les décideurs, et donc les chercheurs, à s'intéresser à la possibilité d'extraire de la connaissance de ces gigantesques mines d'information. Le domaine de la fouille de données a ainsi connu un essor considérable dans le milieu des années 90 avec la conception et le développement d'algorithmes et d'outils efficaces, capables de traiter de grands volumes de données, dans des temps raisonnables, sur des machines aux capacités limitées. Les domaines d'applications ont été variés depuis une quinzaine d'années allant des domaines financiers et marchands, aux domaines de la biologie, de la santé, en passant par les domaines technologiques, etc. La nature des données explorées est également très diversifiée. Celles-ci peuvent, par exemple, être numériques ou symboliques, multimédia ou pas, ensemblistes ou séquentielles, etc. Dans le cadre de cette thèse, **nous nous intéressons aux domaines d'applications**

[1] http://www.wintercorp.com

où les données explorables sont symboliques et séquentielles. Parmi ceux-ci, on peut citer la langue naturelle qui fournit des ensembles de phrases sous la forme de séquences de mots sur lesquelles on cherchera, par exemple, à découvrir des modèles de tournures de phrases utilisées par certaines catégories socioprofessionnelles ou par certaines ethnies, etc. On peut aussi citer, en bioinformatique, l'exploration de séquences de gènes parmi lesquelles on va par exemple chercher à découvrir de la connaissance sur les causes de déclenchement de certains cancers chez des catégories d'individus. Dans le domaine industriel, la modélisation des alarmes dans une installation, en utilisant les historiques des séquences d'actions ayant conduit à une alarme en son sein, est un autre exemple d'applications largement étudiées par la communauté scientifique. Dans le domaine du Web, un certain nombre de travaux très en vogue consistent à étudier les séquences de pages visitées par les internautes d'un site Web donné. Les données sont alors les fichiers logs du site Web considéré, comportant en fait, pour chaque internaute, la séquence des pages qu'il a visitées durant une session de navigation.

La grande majorité des algorithmes de fouille de données, qu'ils soient conçus pour traiter des données ensemblistes ou séquentielles, s'intéressent à la découverte de modèles (ou motifs) fréquents. Dans ce cadre, l'utilisateur doit définir un certain seuil de fréquence d'apparition des motifs et l'objectif de la phase de fouille de données est alors de rechercher, dans un ou plusieurs jeux de données, les motifs apparaissant avec une fréquence supérieure au seuil fixé au départ. Ce type de contrainte élémentaire (la fréquence d'apparition d'un motif), qui peut être vu comme un critère objectif de la qualité des motifs extraits, a été durant de nombreuses années l'objet de toutes les attentions de la part des concepteurs de nouveaux algorithmes de fouille. Dans ce contexte, les critères utilisés pour évaluer ces algorithmes sont la *complétude* et la *correction*. Ainsi, on cherche à vérifier que tous les motifs vérifiant cette contrainte de fréquence soient trouvés (complétude) et que seulement ceux-ci soient trouvés (correction). A noter qu'au fil du temps, d'autres critères objectifs de qualité ont été mis au point, comme le lift [McG05, GH06]. D'autres critères comme la complexité en temps et en espace des algorithmes sont également couramment étudiés. Nous pensons que l'évaluation des algorithmes telle qu'elle est effectuée classiquement avec la complétude et la correction est insuffisante. Considérons l'exemple suivant. On dispose d'un ensemble de données issues de l'observation des naissances dans une maternité. Sur une journée, on observe la naissance de 15 garçons (soit 15 séquences d'un élément <G>) et de 5 filles (soit 5 séquences d'un élément <F>). Sur un seul jour d'observation, cette répartition est tout à fait possible. Si, sur cet échantillon de 20 séquences, on recherche les séquences ayant une fréquence supérieure à un seuil de 40%, alors seule la séquence <G> sera considérée comme fréquente puisqu'elle a une fréquence de 75%. Ce résultat est bien sûr valable au sens de la correction et de la complétude, toutefois, peut-on en conclure que les naissances de garçons sont plus fréquentes que celles de filles? Il semble évident que le fait que l'étude soit menée sur une quantité de données faible a une forte influence sur la réponse à cette question. Toutefois, malgré ce constat évident, il est à noter que la taille de l'échantillon utilisé pour la fouille n'est en général pas prise en compte pour décider de la fréquence d'un motif.

Utiliser simplement l'argument que la fouille de données a pour objectif principal de

traiter des volumes de données importants, et donc que les ensembles de séquences sont généralement grands, n'est pas une justification satisfaisante à cette non prise en compte de la taille. En effet, il est des situations où il n'est possible de disposer que de jeux de données de taille très limitée (en biologie moléculaire par exemple). Quid alors de la possibilité de réaliser une fouille de données pertinente sur de tels jeux ? Que pourra-t-on dire alors de la qualité des motifs extraits ? Il est, d'autre part, des situations où il est possible de disposer de masses de données très importantes mais dans ce cas le processus de fouille peut alors nécessiter un temps de traitement très conséquent et des moyens de calcul significatifs. Une question que l'on est alors en droit de se poser est : que se passerait-il si l'on n'explorait pas entièrement le jeu de données à notre disposition ? Quelle pertinence pourrait-on alors accorder aux motifs que nous pourrions extraire par rapport à ceux que nous aurions pu extraire si nous avions effectué une fouille sur le jeu de données complet ? Selon le domaine d'application, l'écart entre les motifs que l'on pourrait extraire sur le jeu entier et sur une partie seulement de celui-ci peut avoir un impact important. En biologie moléculaire à nouveau, si l'on cherche à modéliser l'effet de certaines molécules sur des patients et que le jeu de données est trop petit, l'expert peut être amené, à partir des motifs extraits (de mauvaise qualité), à tirer des conclusions hâtives pouvant avoir des conséquences dramatiques sur la santé des futurs patients.

En fait, lorsque l'on explore un jeu de données afin de chercher à en extraire un modèle d'une situation quelconque, on travaille sur un sous-ensemble d'une distribution statistique cible et, c'est en fait celle-ci qui est réellement intéressante. Le jeu de données en lui-même n'en est qu'un échantillon. Dans le cadre de cette thèse, pour offrir une meilleure évaluation des résultats d'un processus de fouille **nous nous intéressons à la pertinence statistique des motifs extraits**, en prenant en compte cette distribution théorique, inconnue. Il faut réellement avoir conscience qu'extraire des motifs d'un jeu de données peut conduire à considérer des motifs comme fréquents parce que sur ce jeu, exceptionnellement, ceux-ci étaient effectivement fréquents, mais en fait sur la distribution sous-jacente ils ne l'auraient pas été. De même, sur un jeu donné, à cause des fluctuations d'échantillonnage, un motif peut avoir une fréquence exceptionnellement basse et n'être donc pas considéré comme fréquent alors que sur la distribution statistique sous-jacente de ces données, le motif aurait été réellement fréquent. Reprenant l'exemple des naissances dans un hôpital, sur le long terme, on en viendrait évidemment à observer qu'environ 50% des naissances sont des garçons et que donc le motif <F> a été oublié à tort puisque dans la distribution sous-jacente au jeu de données, ce motif a une fréquence supérieure à 40%.

S'intéresser à la distribution sous-jacente à un jeu de données n'est évidemment pas une tâche triviale puisque celle-ci n'est en général pas connue. La première solution que nous proposons pour résoudre ce problème consiste à utiliser des outils de la statistique inférentielle pour chercher une borne sur le nombre de séquences nécessaires pour limiter les erreurs réalisées en fouillant le jeu de données plutôt que la distribution sous-jacente. Dans le contexte de notre exemple, cela revient à chercher à répondre à la question : combien faut-il de naissances pour que les motifs fréquents obtenus contiennent moins de x% de motifs oubliés et y% de motifs trouvés par chance alors qu'ils n'auraient pas

dû l'être ? Cette borne constitue notre **première contribution** dans le cadre de cette thèse.

Lorsque cette borne n'est pas applicable en pratique, nous proposons de chercher à inférer le langage dont sont issues les données séquentielles que nous souhaitons fouiller. En fait, l'inférence de modèles à partir de jeu de données séquentielles n'est pas l'apanage de la fouille de données. Il existe des techniques issues du domaine de l'apprentissage artificiel qui s'intéressent à ce problème depuis de nombreuses années. L'objectif du domaine de l'apprentissage automatique est de concevoir *des méthodes permettant de construire un modèle de la réalité à partir de données, soit en améliorant un modèle partiel ou moins général, soit en créant complètement le modèle* [CM02]. Les Modèles de Markov Cachés [Rab89], l'Inférence Grammaticale [dlH05], figurent parmi les techniques reconnues visant à construire des modèles de la réalité à partir de données séquentielles. Concernant l'inférence grammaticale, des techniques aujourd'hui classiques d'apprentissage artificiel visent à inférer automatiquement, à partir de séquences de mots ou de lettres, des automates probabilistes modélisant le langage sous-jacent au jeu de données. De ce fait, nous pensons qu'il peut être intéressant de chercher à combiner l'apprentissage automatique et la fouille de données dans un nouveau processus d'extraction de connaissances à partir de données. Étant donné un jeu de données constitué d'un ensemble de séquences, nous proposons d'utiliser des algorithmes d'inférence grammaticale pour en inférer un automate modélisant la distribution sous-jacente. Un nouvel algorithme de fouille, que nous présentons dans ce document, peut alors ensuite chercher à extraire, de l'automate, des motifs supposés pertinents sous certaines conditions. La phase d'apprentissage artificiel permet ainsi de généraliser le jeu de données et de disposer ensuite d'une représentation condensée de l'ensemble de séquences initiales. Le développement d'un tel algorithme de fouille d'automates constitue la **seconde contribution** significative de cette thèse.

L'intégration de contraintes diverses au sein des algorithmes d'extraction de motifs a été l'objet de recherches avancées ces dernières années [BJ05]. L'extraction sous contraintes a pour but de faire diminuer la taille de l'ensemble des motifs, permettant ainsi une meilleure interprétation des résultats, mais aussi d'affiner la recherche sur des critères précis. Dans le contexte de cette thèse, nous proposons également d'intégrer un certain nombre de contraintes dans les algorithmes que nous avons mis au point. Ceci constitue la **troisième contribution** de cette thèse.

Le respect du caractère privé de certaines informations lors du processus d'extraction de connaissance à partir de données est un thème devenu central dans la communauté fouille de données depuis bientôt une dizaine d'années. Comme nous allons le voir dans cette thèse, une **quatrième contribution** consiste à montrer comment l'un des effets de bord intéressant des travaux que nous avons menés est d'offrir un cadre idéal pour des techniques de préservation de la vie privée dans certaines classes d'applications : celles reposant sur les flux de données.

Ce mémoire de thèse est composé de deux grandes parties. La première présente l'état de l'art ainsi que les concepts et notations requis pour la compréhension de la

suite du manuscrit. Elle est divisée en trois chapitres :

- **Chapitre 1** : Il présente le cadre général de l'extraction de connaissances à partir de données puis la fouille de données et plus particulièrement la fouille de données séquentielles. Divers algorithmes sont présentés et différents thèmes sont abordés tels que la fouille de données sous contraintes et la fouille de données préservant la vie privée.
- **Chapitre 2** : Il traite de l'inférence grammaticale, et plus particulièrement de l'inférence grammaticale probabiliste. Les automates probabilistes sont présentés et deux algorithmes utiles à leur apprentissage sont plus particulièrement développés.
- **Chapitre 3** : Il présente la méthode de Hingston [Hin02] permettant d'extraire des motifs et leurs probabilités à partir d'automates probabilistes. Ces travaux constitueront le point de départ de nos contributions sur la fouille d'automates.

La deuxième partie de cette thèse est constituée de deux chapitres qui présentent l'ensemble de nos contributions :

- **Chapitre 4** : Il introduit ce que nous appelons la "fouille de données séquentielles probabiliste", c'est-à-dire la fouille de données séquentielles tenant compte de la distribution statistique sous-jacente aux jeux de données. Nous présentons dans un premier temps une étude sur les diverses erreurs qu'il est possible de commettre en fouille de données classique puis nous proposons une borne théorique sur la taille nécessaire des jeux de données afin d'assurer des résultats avec des taux d'erreurs théoriques fixés à l'avance. Nous présentons ensuite une étude sur la qualité des automates probabilistes permettant de réaliser une phase de fouille de bonne qualité. Cela nous conduit à proposer une seconde borne sur le nombre de symboles dans les séquences permettant d'assurer la construction d'automates pertinents pour l'extraction des probabilités des séquences. Finalement, nous présentons diverses contraintes que nous avons jugé utiles d'intégrer au sein de nos algorithmes de fouille.
- **Chapitre 5** : Il présente une application des algorithmes présentés au chapitre 4 dans le contexte de l'étude du trafic routier. Nous présentons un prototype que nous avons appelé Traffic Miner et montrons que la fouille de données à partir d'automates peut constituer une piste très intéressante dans le contexte de la préservation du caractère privé de certaines informations.

Ce mémoire de thèse se termine sur une conclusion, ouvrant de multiples perspectives de recherche.

Première partie

État de l'art

1 Fouille de données

Sommaire

Résumé

Dans ce chapitre, nous présentons la fouille de données dans son cadre général, puis les concepts plus spécifiques qui nous seront utiles par la suite. Certains algorithmes sont présentés, en particulier l'algorithme SPAM que nous avons utilisé dans le cadre de nos travaux. La fouille de données étant une étape d'un processus visant à extraire des connaissances répondant à des contraintes spécifiques, nous présentons quelques unes de ces contraintes, puis nous proposons un aperçu des travaux étudiant la pertinence ou l'exactitude des connaissances extraites. Enfin, nous présentons un axe récent de la fouille de données qui vise a assurer la préservation du caractère privé des données considérées.

1.1 Introduction

La fouille de données est une étape du processus complexe d'Extraction de Connaissances à partir de Données (ECD). L'ECD a été définie par FRAWLEY ET AL. [FPSM91] comme étant l'extraction non triviale, à partir de données, de motifs valides, nouveaux, potentiellement utiles et compréhensibles. L'extraction de connaissances à partir de données a connu un essor très important depuis le milieu des années 90 de par son potentiel économique significatif. Historiquement, les premiers algorithmes efficaces ayant permis de traiter des volumes de données importants ont été utilisés dans le domaine du marketing, notamment dans le contexte des achats des clients dans les hypermarchés. L'objectif est alors de découvrir, dans les masses énormes d'achats de clients, des associations de produits achetés fréquemment par ceux-ci. Une telle connaissance peut ensuite être utilisée pour agencer judicieusement les magasins, pour mener des campagnes marketing sur des produits spécifiques, etc. Comme on peut l'imaginer, l'impact

économique de tels outils peut ainsi être considérable. Dans le domaine bancaire, l'extraction de connaissances à partir de données est également très utilisée dans le but de proposer des services mieux adaptés aux clients, de détecter des utilisations anormales de moyens de paiement, de définir des comportements fréquents de clients à risque, etc. Ici également, on imagine bien les gains financiers significatifs que de telles techniques peuvent apporter aux banques. Dans le domaine des assurances, la fouille de données a également été utilisée avec succès pour découvrir des comportements frauduleux de groupes de clients.

Au cours des dernières années, les évolutions technologiques ont permis de créer des machines de plus en plus puissantes, avec des capacités de stockage de plus en plus importantes. L'amélioration des techniques de collecte et de sauvegarde a ainsi entraîné la diversification et l'augmentation de la quantité des données susceptibles d'être exploitées. De ce fait, le domaine de l'extraction de connaissances à partir de données est en constante évolution. Toutefois, de façon générale, le processus d'extraction de connaissances à partir de données peut être schématisé par la figure 1.1 [FPSS96].

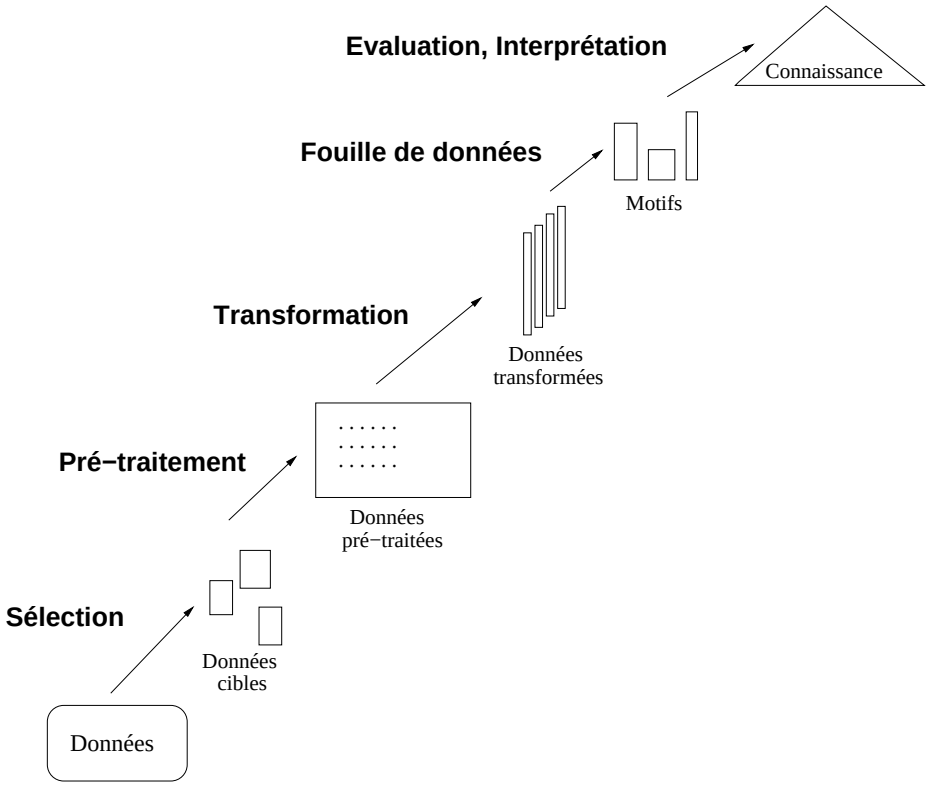

Fig. 1.1 – *Aperçu des étapes du processus d'*ECD.

La première étape consiste à sélectionner un sous-ensemble des données qui peut, par exemple, se limiter à certaines variables, à partir duquel la connaissance doit être extraite. Ensuite, arrive la phase de pré-traitement des données qui permet de nettoyer

les données bruitées ou manquantes. Puis les données sont transformées dans l'objectif d'être adaptées à la tâche de fouille de données. Les meilleurs attributs, en fonction de la tâche à effectuer, sont sélectionnés. Parfois, des méthodes de réduction ou de transformation sont effectuées pour réduire le nombre de variables. La phase suivante est celle de la fouille de données en elle-même, dans laquelle le but précis de la fouille est défini (classification, clustering, régression, etc.). Ensuite, toujours dans cette même phase, les algorithmes (ainsi que leurs paramètres) sont choisis. Ils sont ensuite exécutés sur les données transformées, pour obtenir les motifs intéressants. Nous détaillerons cette étape de la fouille de données par la suite. Enfin, la dernière étape est celle de l'interprétation des résultats qui peut parfois amener à recommencer le processus à son origine, ou à modifier une des étapes.

Il existe un très grand nombre de types de données au sein desquelles il peut être intéressant d'extraire de la connaissance. Les premiers travaux dans le domaine de la fouille de données se sont principalement intéressés aux données structurées stockées dans des bases de données relationnelles. Dans ce contexte, les données sont organisées en un ensemble de tables contenant un certain nombre de colonnes (les attributs) et de lignes (les individus). On peut alors s'intéresser à effectuer une fouille sur les valeurs des attributs quels que soient les individus ; ceci correspond à *la fouille de données transactionnelles*. On peut aussi s'intéresser à l'évolution des valeurs des attributs en fonction du temps pour chaque individu ; ceci décrit le cadre de *la fouille de données séquentielles* qui nous concernera tout au long de ce manuscrit. C'est dans ce dernier contexte qu'ont vu le jour les travaux d'Agrawal en 1995 [AS95]. Étant donnés des achats de clients au cours du temps dans un hypermarché, l'objectif des travaux d'Agrawal était de chercher à extraire les séquences fréquentes d'achats des clients lors de leurs visites successives dans le magasin. En effet, ces études permettent des améliorations pour les groupes de vente, entre autre, cela peut permettre un meilleur approvisionnement des stocks, une meilleure disposition à l'intérieur du magasin pour que les clients soient "contraints" de passer devant un produit spécifique, de proposer des offres personnalisées aux clients, etc. Dans le domaine de la surveillance industrielle, la fouille de données séquentielles peut être utilisée pour découvrir des motifs fréquents de déclenchement d'alarme, ce qui peut aider les responsables lors de leurs recherches de défauts sur les chaînes de production [KMT99]. Plus récemment, dans le domaine de la fouille de données issues du Web [KB00], divers systèmes ont été proposés pour utiliser les logs de sites Web dans le but de modéliser les comportements des internautes [SP01]. Cela peut, entre autre, tendre à l'amélioration de l'ergonomie du site, ou à la création de sites dynamiques permettant d'adapter chaque nouvelle page visitée par le client en fonction des précédentes. Dans un cadre quelque peu différent, Mannila et al. [MTV97] proposent d'étudier la fouille d'épisodes fréquents. Cela consiste à rechercher dans une unique séquence très longue des sous-séquences fréquentes. Il existe aussi des bases de données temporelles. Par exemple, dans le secteur médical on peut disposer de données indiquant la durée des séjours des patients et utiliser celles-ci pour prédire le parcours d'un malade dans les services de l'hôpital et le temps que durera son séjour. Il existe encore beaucoup d'autres types de bases de données (spatiales, multimédia, etc.) que nous ne traiterons pas ici.

Nous présentons dans ce chapitre le cadre général de la fouille de données, appliquée aux données transactionnelles, puis nous développons le cadre plus restrictif de la fouille de données séquentielles. En effet, ces dernières ont la contrainte supplémentaire de devoir respecter un ordre établi, comme par exemple l'ordre temporel.

En fouille de données, l'extraction de connaissances passe par la recherche d'évènements répondant à certaines contraintes dans l'ensemble de données. Lorsqu'on cherche à extraire des motifs, on souhaite qu'ils soient de bonne qualité. De nombreux travaux ont été menés sur le thème de la qualité de la connaissance extraite par des algorithmes de fouille. On peut distinguer principalement deux types de critères de qualité. Les critères *subjectifs* de qualité sont généralement quantifiés par l'être humain (gain financier obtenu par la connaissance extraite dans une banque, nombre de vies sauvées par la connaissance découverte dans le domaine médicale, etc.). Les critères *objectifs* résultent eux de contraintes calculables définies auparavant par l'utilisateur. Les critères de qualité les plus communs dans le cadre, par exemple, de l'extraction de règles d'association à partir de données transactionnelles sont le *support* et la *confiance* [AS94]. Depuis quelques années, d'autres types de contraintes ont été étudiées au sein des algorithmes de fouille de données, donnant naissance à une classe d'algorithmes de fouille dit "sous contraintes" [BJ05].

1.2 Définitions

Comme nous l'avons précisé précédemment, notre étude portera sur des données séquentielles. Néanmoins, le cadre théorique est le même que celui de la fouille de données transactionnelles. Nous présentons donc celui-ci en utilisant des définitions initialement proposées par Agrawal et Srikant dans [AS95].

Définition 1.1 (Item, itemset) *Un ensemble possible* **d'items** *est noté par* Σ. *Un* **itemset** X *est un sous-ensemble de* Σ. X *est constitué d'items* x_1,x_2,...,$x_{|X|}$ *que l'on notera* $(x_1 x_2 \dots x_{|X|})$.

Comme nous l'avons déjà précisé, le cadre le plus classique de découverte d'itemsets fréquents est l'analyse d'ensembles d'achats, plus communément appelé "panier de la ménagère". Dans ce contexte, on définit une transaction de la façon suivante :

Définition 1.2 (Transaction) *Une* **transaction** T *est un triplet* (C,t,X), *où* X *est l'ensemble des items (itemset) achetés par le client* C *à la date* t. *Le couple* (C,t) *constitue un identifiant unique de la transaction* T.

Nous allons illustrer chacune des définitions sur un exemple issu du domaine des achats dans un hypermarché. Le tableau 1.1 contient 12 transactions correspondant à des achats effectués par 4 clients à diverses dates. Il est important de noter qu'un itemset est un ensemble (ensemble des produit achetés par un client à une date donnée) et donc contient des éléments non ordonnés.

Définition 1.3 (Séquence) *Une* **séquence** S, *notée* $< X_1 X_2 \dots X_n >$, *est la liste ordonnée des itemsets* X_j. *Elle représente la suite ordonnée des achats effectués par un*

Client	date	Items
C_1	3	Biscuit, Couches, Fruit
C_1	9	Assouplissant
C_1	13	Eau, Glace
C_2	4	Biscuit, Couches
C_2	5	Détergent
C_2	6	Assouplissant
C_2	11	Eau, Glace
C_3	2	Biscuit, Fruit
C_3	7	Détergent, Glace
C_3	10	Assouplissant
C_4	1	Biscuit, Couches
C_4	8	Détergent, Eau
C_4	12	Assouplissant, Eau

TAB. 1.1 – *Base de données constituée de 12 transactions.*

client au cours du temps. On notera $|S|$ la longueur de la séquence S, c'est-à-dire le nombre d'itemsets qui la composent.

Le tableau 1.2 représente les séquences construites à partir du tableau 1.1. Pour plus de clarté, nous remplaçons chaque produit par son initiale.

Client	Séquences
C_1	<(B C F) (A) (E G)>
C_2	<(B C) (D) (A) (E G)>
C_3	<(B F) (D G) (A)>
C_4	<(B C) (D E) (A E)>

TAB. 1.2 – *Base de séquences.*

Exemple 1 *Dans le tableau 1.2, l'ensemble des items est $\Sigma = \{A,B,C,D,E,F,G\}$, (BCF) est un itemset, $T = (C_3,1,(BF))$ est une transaction et $S =< (BC)(DE)(AE) >$ est la séquence correspondant au client C_4. $|S|$ est égal à 3.*

Définition 1.4 (Sous-séquence) *Une séquence $S_1 =< X_1X_2 \ldots X_n >$ est* **incluse** *dans une séquence $S_2 =< Y_1Y_2 \ldots Y_m >$ (noté $S_1 \prec S_2$) si et seulement si il existe des entiers $i_1 < i_2 < \ldots < i_n$ tels que $X_1 \subseteq Y_{i_1}, X_2 \subseteq Y_{i_2}, \ldots, X_n \subseteq Y_{i_n}$. On dit que S_1 est une* **sous-séquence** *de S_2.*

On utilise aussi souvent le terme de **motif** pour désigner une sous-séquence.

Définition 1.5 *Un client* **supporte** *une séquence S si celle-ci est incluse dans sa séquence de données .*

Pour la table 1.2, nous noterons S_i la séquence correspondant au client C_i.

Exemple 2 *La séquence* $S =< (BC)(E) >$ *est incluse dans la séquence* $S_4 =< (BC)(DE)(AE) >$. *On dira donc que le client* C_4 *supporte* S. *Au contraire, la séquence* $S' =< (BCE)(A) >$ *n'est pas supportée par le client* C_4.

Définition 1.6 (Support) *Le* **support** *d'une séquence* S *dans une base de données* DB *de séquences, noté* $supp(S,DB)$, *est égal à la proportion de clients qui supportent* S *dans la base* DB.

Lorsque la base de données DB est implicite et qu'il n'est pas nécessaire de préciser, on note le support de S par $supp(S)$.

Lors du calcul du support d'une sous-séquence, il est important de noter que les séquences, dans lesquelles elle est incluse, ne sont prises en compte qu'une seule fois, même si elle y apparaît de plusieurs façons différentes.

Exemple 3 *Le support de la séquence* $S =< (B)(D)(A) >$ *est égal à* $\frac{3}{4}$ *car elle est supportée par les clients* C_2, C_3 *et* C_4. *Le support de la séquence* $S' =< (B)(E) >$ *est aussi de* $\frac{3}{4}$ *même si cette séquence apparaît deux fois dans* S_4.

Définition 1.7 (Séquence maximale) *Une séquence* $S =< X_1X_2 \dots X_n >$ *est* **maximale** *dans un ensemble de séquences* L *si et seulement si il n'existe pas* $S' \in L$ *tel que* $S \prec S'$.

Exemple 4 *La séquence* $< (BC)(D)(A)(EG) >$ *est maximale pour l'ensemble de séquences de la table 1.2. La séquence* $< (BF)(D)(A) >$ *n'est pas maximale car incluse dans* S_3.

Propriété 1.1 *Soient* S_1 *et* S_2 *deux séquences. Si* $S_1 \prec S_2$ *alors* $supp(S_1) \geq supp(S_2)$.

1.2.1 Cas particulier

Il existe des domaines où les données séquentielles sont constituées de séquences dans lesquelles tous les itemsets sont de taille 1. C'est le cas, par exemple, des données séquentielles issues des fichiers logs de sites Web, des séquences d'ADN en génomique, des phrases d'une langue naturelle, etc. Il est évident que dans ce contexte toutes les définitions précédentes s'appliquent également.

Pour alléger les notations, nous noterons $< x_1x_2 \dots x_n >$ la séquence $< (x_1)(x_2) \dots (x_n) >$ où tous les itemsets sont de taille 1. Si l'on considère le domaine de la génomique par exemple, le tableau 1.3 présente un ensemble de séquences d'ADN.

L'ensemble des items est ici $\Sigma = \{A,C,G,T\}$ et $< ACGT >$ est une séquence de longueur 4. La séquence $< AGT >$ est incluse dans toutes les séquences du tableau 1.3, son support est donc de 1 ($\frac{4}{4}$).

Id	Séquences
1	<ACGT>
2	<ATGAT>
3	<CAGTA>
4	<AGGATT>

TAB. 1.3 – *Base de séquences d'ADN.*

1.3 Les algorithmes de fouille de données séquentielles

Historiquement, les premiers algorithmes de recherche de motifs séquentiels fréquents ont été proposés par AGRAWAL ET SRIKANT dans [AS95]. Ces algorithmes se nomment APRIORIALL et APRIORISOME, et sont inspirés de l'algorithme APRIORI de recherche d'itemsets fréquents proposé dans [AS94]. Nous présentons en détails l'algorithme APRIORIALL par la suite. La plupart des algorithmes de recherche de sous-séquences fréquentes sont basés sur deux phases distinctes réitérées : *(i)* la génération des séquences candidates, puis *(ii)* le filtrage grâce à un certain nombre de contraintes. Dans le cas de l'extraction de motifs fréquents, le filtrage se fait grâce au seuil de support.

Parmi les algorithmes de recherche de motifs fréquents, nous trouvons deux stratégies de recherche différentes : la recherche en *profondeur d'abord* et celle en *largeur d'abord*, appelée aussi *par niveau*. Les algorithmes basés sur une recherche en profondeur utilisent généralement des arbres de préfixes (par exemple les algorithmes PSP [MCP98], FREESPAN [HPMA+00] ou PREFIXSPAN [PHP+01]), ou des structures similaires, qu'ils explorent pour construire les candidats. Le principe ici est de construire une nouvelle représentation de la base de données avec une indexation par les préfixes. Dans les algorithmes de recherche par niveau, on recherche les motifs en augmentant leurs tailles à chaque étape. La base de données est généralement utilisée telle quelle et les motifs découverts à chaque étape sont utilisés pour générer des candidats de longueurs supérieures à l'étape suivante. Dans le domaine des algorithmes par niveau, on peut citer notamment GSP [SA96] et SPADE [Zak01]. Pour une étude détaillée des algorithmes de recherche de motifs séquentiels, nous renvoyons le lecteur intéressé vers [MTP04].

L'algorithme GSP [SA96] est une extension de APRIORIALL qui propose la prise en compte d'intervalles de temps entre les items. De plus, il permet de prendre en compte des taxinomies qui permettent de classer les items dans des catégories d'appartenance. Ceci donne la possibilité de rechercher des règles de plus hauts niveaux portant sur des ensembles d'items (des genres de films par exemple) plutôt que des items particuliers (des titres de films). GSP utilise un arbre de hachage pour classer les motifs en fonction de leurs préfixes et donc faciliter leur recherche.

L'algorithme SPADE [Zak01] propose des améliorations par rapport à GSP, principalement dans la phase de génération des candidats. En effet, il utilise des classes d'équivalence pour gérer les candidats et les séquences. Deux éléments de longueur k

appartiendront à la même classe s'ils ont un préfixe commun de longueur $k-1$. Différents types de jointures sont ensuite utilisés sur les classes d'équivalence pour générer les candidats. De plus, l'algorithme utilise une représentation horizontale de la base de données, où l'on répertorie, pour chaque item, l'ensemble des couples *(identifiant, itemset)* qui le contiennent.

Il existe également d'autres méthodes qui permettent une recherche incrémentale des motifs lorsque des éléments sont ajoutés à la base de données (voir [MPT03], [PZOD99] et [ZXML02]).

Il existe de nombreux autres algorithmes de recherche de sous-séquences fréquentes qui répondent à des problématiques spécifiques. Yun (voir [Yun08]) présente par exemple une méthode permettant d'associer des poids aux items selon leur importance. En effet, dans des données issues du Web ou des données ADN, chaque item n'a pas la même importance. Yun donne l'exemple de l'étude du panier de la ménagère. Lorsque le volume de données à traiter est trop important, il est souvent nécessaire, avec les algorithmes classiques, de choisir un support élevé. Dans ce cas, les objets coûteux (qui sont donc peu fréquents, mais non moins intéressants) ne vont pas apparaître dans les sous-séquences fréquentes. Dans ce contexte, les poids à affecter aux items pourront être assimilés aux prix des objets. Dans son algorithme WSpan, Yun propose ainsi d'intégrer des poids dans le calcul du support des séquences. Des algorithmes intégrant cette notion de poids sur les items avaient jusqu'ici plutôt été développés dans le cadre de la recherche de règles d'associations (voir [YL05], [TMF03], [CFCK98]).

D'autres types d'algorithmes visent à extraire des motifs non exacts, ils utilisent des critères de similarité entre les séquences. Ces méthodes peuvent être utiles en génomique où il est connu que les séquences peuvent subir des mutations. Dans [ZYHY07], les auteurs proposent d'utiliser la distance de Hamming entre les séquences, ils calculent donc le support d'un groupe de séquences dont les distances sont inférieures à un certain seuil (dépendant de la longueur des séquences). Dans [ALB03b], les auteurs proposent d'utiliser des expressions régulières comme contrainte de similarité qu'ils représentent sous la forme d'arbres.

On constate qu'il existe à ce jour un grand nombre d'algorithme d'extraction de motifs séquentiels. Chacun d'eux est plus ou moins bien adapté aux données à traiter dans les diverses applications spécifiques où il peut être fait appel à eux. Le choix d'un algorithme de fouille de données séquentielles particulier sera ainsi, en général, dicté par la longueur des séquences à traiter, la taille de l'ensemble des items, le taux de redondance dans les données, les types de contraintes que l'on souhaite imposer sur les données à traiter ou les motifs à extraire, etc. Dans la suite de cette section, nous détaillons maintenant deux algorithmes typiques de la fouille de données séquentielles.

1.3.1 AprioriAll : un algorithme de recherche par niveau

Agrawal et Srikant ont été les premiers à proposer des algorithmes de recherche de motifs fréquents. Dans [AS95], Agrawal propose l'algorithme AprioriAll qui recherche toutes les séquences fréquentes et l'algorithme AprioriSome qui recherche uniquement les séquences fréquentes maximales (voir la définition 1.7).

L'algorithme AprioriAll est basé sur deux phases :

– La première phase vise à générer des séquences candidates au statut de séquences fréquentes (voir l'algorithme 1.1). A chaque étape k de l'algorithme, les séquences fréquentes extraites à l'étape $k-1$ (de longueur $k-1$) sont utilisées pour calculer les nouvelles séquences candidates. Une séquence candidate à l'étape k ne contient que des sous-séquences fréquentes de l'étape $k-1$. Cette construction est basée sur la propriété 1.1 qui montre que la contrainte de fréquence est une contrainte anti-monotone (voir la section 1.4.1). En effet, si une sous-séquence $S^{'}$ de la séquence candidate S n'est pas fréquente (n'apparaît pas dans C_{k-1}) alors la séquence S ne pourra pas être fréquente. Pour la génération des candidats de taille 1, tous les items de l'alphabet sont utilisés.

Algorithme 1.1 : Algorithme AprioriGen

Entrées : L_{k-1} l'ensemble des séquences fréquentes de longueur $k-1$
Sorties : C_k l'ensemble des séquences candidates
début
  $C_k \leftarrow \emptyset$;
  pour toutes les *séquences* $S_1 =< X_1 \dots X_{k-1} > \in L_{k-1}$,
  $S_2 =< Y_1 \dots Y_{k-1} > \in L_{k-1}$ **faire**
    si $X_1 = Y_1$, $X_2 = Y_2$, …, $X_{k-2} = Y_{k-2}$ **alors**
      $C_k \leftarrow < X_1, \dots, X_{k-1}, Y_{k-1} >$;
  pour toutes les *séquences candidates* $C \in C_k$ **faire**
    pour toutes les *séquences* D *tel que* $|D| = k-1$ *et* $D \prec C$ **faire**
      si $D \notin L_{k-1}$ **alors**
        supprimer C de C_k ;
  retourner C_k;
fin

– La seconde phase, le filtrage des candidats, consiste en une phase de calcul du support de chaque séquence candidate. Pour tous les motifs candidats calculés dans la phase de génération, le support est calculé et toutes les séquences dépassant le seuil de support, donné en paramètre par l'utilisateur, sont conservées comme motifs fréquents (voir l'algorithme 1.2).

Pour illustrer cet algorithme, considérons à nouveau les séquences de la table 1.2 (reprises au tableau 1.4) et supposons que *min_supp* soit fixé à 0.75.

Supposons maintenant que l'on ait calculé les motifs de L_2, c'est-à-dire les sous-séquences fréquentes de longueur 2 et que l'on cherche à calculer les motifs de L_3. Les candidats de longueur 3 sont construits à partir de L_2. Le tableau 1.5 présente les résultats des calculs des algorithmes 1.1 et 1.2. La deuxième colonne du tableau correspond aux motifs issus de la jointure des motifs fréquents de longueur 2, ceux-ci sont construits par la première boucle de l'algorithme 1.1. La troisième colonne est le résultat après suppression des candidats contenant des sous-séquences non fréquentes, c'est-à-dire après

Algorithme 1.2 : Algorithme AprioriAll

Entrées : L_1 l'ensemble des séquences fréquentes de longueur 1, *min_supp* le support minimum, une base de séquences DB.
Sorties : L l'ensemble des séquences fréquentes
début
  pour *($k = 2$; $L_{k-1} \neq \emptyset$; $k++$)* **faire**
    $C_k = Apriori\_Gen(L_{k-1})$;
    pour toutes les *séquences candidates* $C \in C_k$ **faire**
      $supp(C) \leftarrow 0$;
    pour toutes les *séquences* $S \in DB$ **faire**
      pour toutes les *séquences candidates* $C \in C_k$ **faire**
        si $C \prec S$ **alors**
          $supp(C) \leftarrow supp(C) + \frac{1}{|DB|}$;
    pour toutes les *séquences candidates* $C \in C_k$ **faire**
      si $supp(C) > min\_supp$ **alors**
        $L_k \leftarrow C$;
  retourner $L = \cup_i \ L_i$;
fin

Client	Séquences
C_1	<(B C F) (A) (E G)>
C_2	<(B C) (D) (A) (E G)>
C_3	<(B F) (D G) (A)>
C_4	<(B C) (D E) (A E)>

Tab. 1.4 – *Base de séquences.*

la deuxième boucle de l'algorithme 1.1. Par exemple, le motif $(B)(A)(D)$ appartient à la jointure car il a été construit à partir de $(B)(A)$ et $(B)(D)$ mais n'appartient pas à C_3 car la sous-séquence $(A)(D)$ n'est pas incluse dans L_2. Inversement, $(BC)(A)$ appartient à C_3 car toutes les sous-séquences (BC), $(C)(A)$ et $(B)(A)$ sont incluses dans L_2. Ensuite, les supports des trois candidats sont calculés par l'algorithme 1.2 et ils sont tous supérieurs ou égaux au seuil de support. L'ensemble des motifs fréquents de longueur 3 (ensemble L_3) est présenté dans la dernière colonne du tableau.

Notons qu'il existe une autre étape dans l'algorithme AprioriAll, que nous ne présentons pas, qui est une phase de transformation des transactions. En effet, chaque transaction est remplacée par l'ensemble de tous les itemsets fréquents qu'elle contient.

Le principe de cet algorithme est à la base de tous les autres algorithmes de calcul de motifs séquentiels fréquents.

L_2	Jointure	C_3	L_3
(B) (A), $supp = 1$	(B) (A) (D)	non	
	(B) (A) (E)	non	
	(B) (A) (G)	non	
(BC), $supp = \frac{3}{4}$	(BC) (A)	(BC) (A)	(BC) (A), $supp = \frac{3}{4}$
	(BC) (D)	non	
	(BC) (E)	(BC) (E)	(BC) (E), $supp = \frac{3}{4}$
	(BC) (G)	non	
(B) (D), $supp = \frac{3}{4}$	(B) (D) (A)	(B) (D) (A)	(B) (D) (A), $supp = \frac{3}{4}$
	(B) (D) (E)	non	
	(B) (D) (G)	non	
(B) (E), $supp = \frac{3}{4}$	(B) (E) (A)	non	
	(B) (E) (D)	non	
	(B) (E) (G)	non	
(B) (G), $supp = \frac{3}{4}$	(B) (G) (A)	non	
	(B) (G) (D)	non	
	(B) (G) (E)	non	
(C) (A), $supp = \frac{3}{4}$	(C) (A) (E)	non	
(C) (E), $supp = \frac{3}{4}$	(C) (E) (A)	non	
(D) (A), $supp = \frac{3}{4}$			

TAB. 1.5 – *Calcul des motifs fréquents de longueur* 3.

1.3.2 SPAM : un algorithme de recherche en profondeur

SPAM [AFGY02] (Sequential PAttern Mining) est lui aussi un algorithme de recherche de motifs séquentiels fréquents. Contrairement à APRIORIALL qui utilise une stratégie de recherche par niveau, SPAM utilise une stratégie de recherche en profondeur. Sa particularité est qu'il construit une représentation particulière de la base de données, lui permettant un calcul des supports beaucoup plus efficace. Un arbre lexicographique est utilisé pour générer les séquences candidates. En effet, les candidats sont classés, selon un ordre prédéfini, dans un arbre de recherche. Les fils d'une séquence sont générés, par augmentation, durant deux étapes différentes : premièrement une augmentation au niveau de la séquence (*S-Step*) puis une augmentation au niveau des itemsets (*I-Step*). Par exemple, la séquence $< (A)(A) >$ aura pour fils $< (A)(A)(A) >$ et $< (A)(A)(B) >$ générés par *S-Step* et $< (A)(AB) >$ généré par *I-Step*. L'idée majeure implantée dans SPAM consiste à utiliser des bitmaps pour représenter les séquences de la base. La présence et l'absence des items dans les transactions sont codées par des 1 et des 0. Cette représentation binaire permet de construire rapidement des bitmaps pour chaque candidat et de calculer rapidement le support, sans avoir à analyser à nouveau la base de données à chaque nouvelle génération de candidats. Nous donnons, dans le tableau 1.6, un exemple de la représentation utilisée par SPAM pour coder les données. Chaque item est représenté par un bitmap, avec un bit pour chacune des transactions de la base. Nous reprenons les transactions présentées dans le tableau 1.1 et codons chaque

transaction en notant la présence ou l'absence de tous les items.

Client	date	A	B	C	D	E	F	G
C_1	3	0	1	1	0	0	1	0
C_1	9	1	0	0	0	0	0	0
C_1	13	0	0	0	0	1	0	1
C_2	4	0	1	1	0	0	0	0
C_2	5	0	0	0	1	0	0	0
C_2	6	1	0	0	0	0	0	0
C_2	11	0	0	0	0	1	0	1
...	...							

Tab. 1.6 – *Représentation de la table 1.1 sous forme de bitmaps.*

Ce codage permet, grâce à des opérations booléennes, telles que AND et OR, de calculer les bitmaps des séquences candidates, ce qui permet de savoir très rapidement si une séquence est incluse dans une autre et donc de calculer efficacement les valeurs des supports. Par exemple, si l'on considère le processus de génération de candidats *S-Step*, il faut d'abord construire un bitmap intermédiaire de la séquence de départ. Ce bitmap est construit de telle manière que, pour chaque séquence, le premier bit à 1 passe à zéro et tous les suivants passent à 1. Ce bitmap transformé subit l'opération AND avec le bitmap correspondant à l'item à rajouter. Ce processus est illustré dans le tableau 1.7 pour la séquence $< (B)(E) >$ construite à partir de $< (B) >$.

B	B transformé	E	<(B) (E)>
1	0	0	0
0	1	0	0
0	1	1	1
1	0	0	0
0	1	0	0
0	1	0	0
0	1	1	1

Tab. 1.7 – *Processus S-Step pour le calcul du bitmap de* $< (B)(E) >$.

Le principal inconvénient de cet algorithme est qu'il ne supporte pas des séquences de tailles trop importantes du fait de la représentation en bitmaps, ce qui peut être pénalisant dans le contexte des hypermarchés par exemple, où le nombre d'items est de plusieurs milliers. Il est cependant très rapide lorsque l'on ne dépasse pas les limitations imposées dans l'implémentation de SPAM[1], soit pas plus de 64 items par transactions. Pour cette raison, nous avons utilisé cet algorithme pour nos expérimentations.

1 http://himalaya-tools.sourceforge.net/Spam/

1.4 Extraction de motifs sous contraintes

Les algorithmes de recherche de motifs séquentiels proposent de plus en plus la possibilité d'imposer des contraintes autres que la classique contrainte de support [BJ05]. En effet, l'utilisation de la seule contrainte de fréquence peut parfois conduire à générer un nombre de motifs très élevé. Il est alors difficile, voire impossible, de trouver la connaissance pertinente, utile à l'utilisateur, dans cette masse de motifs trop importante. De plus, d'autres contraintes peuvent servir à extraire de la connaissance répondant à des contraintes bien plus spécifiques que le fait d'apparaître un grand nombre de fois dans les données. Il est possible d'imaginer une grande diversité de contraintes, elles peuvent être syntaxiques, sémantiques, temporelles, etc. Pour répondre aux demandes des analystes, différents types de contraintes ont été introduits au cours du temps. En nous inspirant de [PHW02], nous pouvons classer les différents types de contraintes selon les catégories suivantes :

- **Contraintes portant sur les items**. Elles définissent le ou les items qui doivent, ou non, apparaître dans les sous-séquences fréquentes. Par exemple, un analyste d'une banque peut vouloir trouver les relevés de banque des personnes achetant des produits via Internet, pour savoir s'il peut être intéressant de proposer un produit spécifique pour le paiement sécurisé à distance.

- **Contraintes spécifiant la longueur** des motifs ou le nombre d'items par itemset. Ces contraintes de longueur peuvent être maximales, minimales ou exactes. En bio-informatique, elles sont utiles pour prendre en compte les connaissances a priori des biologistes. En effet, ceux-ci peuvent, par exemple, vouloir spécifier à l'analyste la longueur moyenne des séquences d'ADN codant une protéine donnée.

- **Contraintes d'appartenance** de sous-motifs imposant qu'un ensemble de motifs soit contenu dans les motifs recherchés par l'algorithme. Dans une librairie, par exemple, l'analyste peut être intéressé par les individus qui ont acheté les deux premiers livres d'une saga. Peut-être pourra-t-il alors proposer une offre promotionnelle pour le dernier livre de la trilogie.

- **Contraintes d'agrégat** où la fonction d'agrégat peut être une somme, une moyenne, un maximum, un minimum sur les valeurs des items. Dans les applications bancaires de telles contraintes sont particulièrement utiles.

- **Contraintes syntaxiques** basées sur des expressions rationnelles (ou régulières). Elles peuvent permettre d'obtenir des motifs fréquents contenant des expressions particulières. Les expressions régulières ont un pouvoir d'expression plus important que de simple contraintes d'appartenance de sous-motifs.

- **Contraintes temporelles**, mesurant les temps écoulés entre des éléments des séquences. Bien entendu, de telles contraintes sont utiles si l'on dispose d'infor-

mations temporelles. Si l'on considère des suites de symptômes ou d'interventions médicales sur des patients, ces contraintes peuvent être utiles pour s'assurer que les évènements soient proches dans le temps et aient donc un lien de causalité. On pourrait par exemple imposer que chaque item des motifs obtenus concerne des évènements qui se sont déroulés sur une période d'au plus une semaine.

– **Contraintes de distance** entre les itemsets. Elles peuvent imposer des distances maximales ou minimales. Un administrateur de sites Web peut, par exemple, vouloir savoir si les internautes ont visité moins de 10 pages pour passer de la page d'accueil à la page de vente.

– **Contraintes sur les classes** des éléments des séquences (dans le cadre où on travaille avec des taxonomies). Elles sont fréquentes dans les applications du type "panier de la ménagère" pour obtenir des motifs comprenant des éléments appartenant à une unique catégorie.

– **Contraintes de similarité** basées sur des distances d'édition (voir [CMB02]). Elles servent par exemple en biologie où il est connu que les séquences d'ADN connaissent des mutations et il peut parfois être nécessaire de rechercher des motifs non-exacts.

De telles contraintes sont implantées dans divers algorithmes. Par exemple, dans [Zak00], Zaki présente l'algorithme cSPADE qui inclut des contraintes syntaxiques. Il propose différents types de contraintes :

– de longueur ou de largeur,
– de distance minimum ou maximum entre les éléments,
– d'appartenance d'items,
– de fenêtre de temps,
– de classes (lorsque les données appartiennent à des classes).

Pour le calcul des supports des sous-séquences, il utilise une représentation *verticale* de la base de données, c'est-à-dire qu'à chaque item est associée la liste des transactions dans lesquelles il apparaît. Cela permet de calculer le support des séquences en faisant des jointures des ensembles correspondant aux items qui la composent.

[GRS02] présente l'algorithme spirit (Sequential Pattern mIning with Regular expressIon consTraint) décliné en cinq versions différentes. Les auteurs utilisent des expressions régulières, sous la forme d'automates, pour contraindre la recherche de motifs séquentiels fréquents. Les expressions régulières ont l'avantage de présenter une syntaxe simple et compréhensible qui possède un très grand pouvoir d'expression. Les algorithmes spirit implémentent des relaxations de contraintes dans le but d'obtenir de bonnes propriétés d'anti-monotonie (voir la section 1.4.1). Dans [ALB03a], une nouvelle technique pour introduire des contraintes sous forme d'expressions régulières est présentée. Contrairement à l'algorithme spirit, les expressions régulières sont utili-

sées sous forme d'arbres, ce qui permet un traitement plus automatisé des contraintes. L'arbre représentant la contrainte est élagué au cours de la recherche.

Enfin, GSP [SA96] inclut des taxonomies, ce qui permet d'imposer des contraintes sur les catégories des items.

1.4.1 Monotonie des contraintes

L'utilisation de contraintes, telles que celles présentées précédemment, nécessite que celles-ci aient de bonnes propriétés. En fait, les contraintes sont généralement caractérisées par des propriétés de monotonie et d'anti-monotonie.

Propriété 1.2 *Si une contrainte C est* ANTI-MONOTONE *alors si une séquence S satisfait C alors toutes les séquences $S^{'}$ telles que $S^{'} \prec S$ satisfont C.*

Propriété 1.3 *Si une contrainte C est* MONOTONE *alors si une séquence S satisfait C alors toutes les séquences $S^{'}$ telles que $S \prec S^{'}$ satisfont C.*

Les contraintes anti-monotones permettent un élagage important lors de la construction des séquences candidates. En effet, si un motif ne satisfait pas une contrainte anti-monotone, il sera inutile de considérer toutes les séquences le contenant dans les phases de génération suivante. La contrainte de préfixe est une contrainte anti-monotone, du fait de la propriété 1.1, et l'algorithme de génération de AprioriAll (voir l'algorithme 1.1) est basé sur cette propriété. Lorsque les contraintes ne sont pas anti-monotones, le calcul des candidats est beaucoup plus délicat. Il est nécessaire, dans ce cas de relâcher les contraintes. C'est le cas par exemple dans [GRS02] car les contraintes exprimées sous la forme d'expressions régulières ne sont pas anti-monotones.

Nous ne traiterons pas ici de contraintes monotones, mais elles sont toutefois utilisées pour limiter l'espace de recherche. Les contraintes d'agrégats, comme le maximum ou la moyenne peuvent, entre autres, être des contraintes monotones. Ce type de contraintes est, par exemple, étudié dans [PH00]. Dans [BGKW02], on trouve la description de l'algorithme DualMiner qui utilise conjointement des contraintes monotones et anti-monotones.

1.5 Fouille de données et préservation de la vie privée

1.5.1 Présentation

La fouille de données préservant la vie privée [VBF+04] est un nouvel axe de recherche apparu à la fin des années 90. Il regroupe une grande diversité de méthodes qui permettent d'extraire de la connaissance utile à partir de données sensibles, sans pour autant utiliser ou dévoiler des informations privées. Il est difficile de donner une définition très précise de ce que peut être la fouille de données préservant la vie privée car la notion de vie privée est en soit difficilement définissable, ou du moins dépend grandement de l'application concernée. Dans de nombreux domaines peuvent se poser des problèmes de "sensibilité" des données. Clifton et al. proposent dans [CKV+03]

un exemple concernant le domaine médical. Les données que possèdent les centres de sécurité sociale ou de mutuelle pourraient, entre autres, être très utiles pour prédire la progression d'une épidémie. Mais ces instituts ne sont pas forcément enclins à fournir les informations nécessaires, car elles violeraient alors la vie privée de leurs clients.

La sensibilité des données dans un processus de fouille peut se situer à différents niveaux. Tout d'abord, figurent souvent, dans les bases de données, des attributs dont les valeurs doivent être protégées. Nous avons considéré en introduction l'exemple du domaine bancaire. Dans ce contexte, il n'est pas envisageable que les numéros de comptes, les noms ou même les adresses des clients de la banque soient accessibles sans aucune précaution dans le cadre d'un processus de fouille. Pourtant, ces informations peuvent être utiles pour déterminer des caractéristiques sur certaines catégories de personnes, toute discrimination mise à part. Il ne serait donc pas judicieux de simplement supprimer les attributs pouvant poser problème. Des informations sensibles peuvent également être révélées au cours du processus de fouille. Imaginons que nous recherchons des sous-séquences fréquentes dans un ensemble d'itinéraires dans une ville. Si des motifs fréquents liant un quartier résidentiel à d'autres parties de la ville sont découverts, il sera possible, par des moyens externes de retrouver les identités des individus habitants dans ce quartier.

Le challenge des méthodes de fouille préservant le caractère privé de l'information est donc de trouver des modèles efficaces n'accédant pas directement aux informations sensibles des enregistrements qui ne sont pas destinées à être dévoilées. De plus, il ne faut pas que les résultats obtenus par les algorithmes de fouille de données permettent de retrouver, grâce à des moyens adaptés, des informations privées.

Il paraît très difficile de trouver une méthode générique qui permette de protéger à la fois les valeurs des attributs, des relations entre les individus et des connaissances intrinsèques. De plus, comme nous avons pu le voir précédemment, il existe un grand nombre d'algorithmes de fouille de données. Est-il possible alors de trouver des méthodes qui s'appliquent à n'importe quel algorithme ? Le paragraphe suivant expose différentes techniques existantes.

1.5.2 Techniques de préservation de vie privée

Dans cette section, nous évoquons rapidement les techniques pouvant être mises en œuvre dans le cadre de la fouille de données préservant la vie privée. Le lecteur intéressé par de plus amples détails peut se référer à [VCZ06, AY08, BMS08] . Il est important de noter que ces méthodes sont principalement basées sur des heuristiques car le problème de la préservation des données est en fait un problème NP-difficile (voir [ABE+99]), où le problème à résoudre est le suivant : soit D une base de données, R l'ensemble des règles qui peuvent être inférées depuis D et R_h un ensemble de règles inclus dans R. Comment transformer D en $D^{'}$ de telle façon que $R - R_h$ soit obtenu en fouillant $D^{'}$?

Dans [VBF+04], les auteurs proposent le classement des approches de fouille préservant la vie privée en cinq catégories, selon les techniques appliquées. Notons que même si cette classification est discutable, nous avons pris l'option de la reprendre telle

quelle car [VBF+04] constitue une référence majeure en préservation de la vie privée. Les cinq catégories sont donc les suivantes :

- **Distribution des données** : les données peuvent être distribuées horizontalement, ce qui signifie que les enregistrements des bases de données ne se trouvent pas physiquement au même endroit. Par exemple, les transactions sont dispersées sur n sites différents (voir [KC04]). Elle peuvent aussi être distribuées verticalement, c'est-à-dire que toutes les valeurs des différents attributs sont distribuées. Pour les règles d'association, par exemple, les items sont distribués sur plusieurs sites (voir [VC02]). Sur chaque site on ne connaît donc qu'une seule caractéristique de chaque individu. Dans [CKV+03], les auteurs présentent différentes méthodes pour traiter les données partagées (horizontalement ou verticalement) sans dévoiler les informations contenues sur chaque site. Entre autres, ils exposent une méthode très utilisée en cryptographie [Sch96], permettant le calcul sécurisé d'une somme d'éléments contenus sur différents sites, de manière à ce que les valeurs individuelles ne soient pas identifiables.

- **Modification des données** : les données sensibles sont modifiées selon différentes techniques telles que :
  - la perturbation : les valeurs des attributs sont altérées, ce que l'on pourrait assimiler à l'introduction de bruit. Par exemple, dans [AS00], les auteurs proposent de modifier les valeurs des attributs en ajoutant une valeur de distortion (issue d'une loi uniforme ou gaussienne). Les données modifiées sont ensuite utilisées pour estimer la distribution originale des données. En effet, les lois de distortion étant connues, ils proposent une méthode pour estimer la distribution originale, ce qui est différent du fait de reconstruire les données en elles-mêmes.
  - le blocage : les valeurs des attributs sont rendues inaccessibles, notamment en les remplaçant par des caractères jokers neutres (un point d'interrogation par exemple) (voir [CM00] et [SVC01]). Cette approche est parfois plus adaptée que d'inclure du bruit, c'est, par exemple, le cas pour les données médicales.
  - le clustering : les valeurs des attributs sont remplacées par une étiquette plus générale. Dans [AS00], les auteurs utilisent aussi cette méthode pour modifier les données. Des classes sont définies au préalable sur les attributs. Les données sont ensuite utilisées pour estimer la distribution.
  - le swapping : les valeurs de plusieurs enregistrements sont échangées.
  - le sampling : seulement une partie des données est utilisée pour la fouille de données.

  La plupart des méthodes de modification présentées ci-dessus nécessitent un ajustement des valeurs de support ou de confiance. Parfois celles-ci seront remplacées par des intervalles plutôt que des valeurs fixes. En effet, en perturbant la distribution des données les fréquences des motifs seront modifiées.

– **Développements spécifiques aux algorithmes** : contrairement aux autres méthodes, ici il est question d'adapter spécifiquement la méthode de préservation à l'algorithme qui sera ensuite utilisé.

– **Dissimulation de données ou de règles** : des méthodes heuristiques, sous forme de règles de "confusion", sont utilisées pour préserver les données sensibles.

– **Préservation de l'intimité** : une modification sélective des données est opérée, dans le but de garder une utilité dans les résultats obtenus, c'est-à-dire par exemple de conserver le maximum de règles non sensibles. Les méthodes utilisées sont notamment des techniques heuristiques adaptatives, des techniques cryptographiques ou bien des méthodes de reconstruction à partir de données aléatoires. Par exemple, des méthodes cryptographiques ont ainsi été introduites pour résoudre les situations où deux parties (ou plus) veulent accomplir une tâche de fouille de données en mutualisant leurs données mais en ne dévoilant jamais leurs propres données [DA01].

1.5.3 Évaluation des techniques

Comme dans beaucoup de cas, l'évaluation des techniques préservant la vie privée nécessiterait des "benchmarks". Malheureusement, force est de constater que ceux-ci manquent encore à l'heure actuelle.

Les évaluations se font donc selon les critères suivants :

– *La performance* mesurée en terme de temps, de nombre d'opérations, de complexité.
– *L'utilité des données* qui permet de mesurer la perte d'information (d'utilité) des données après application de techniques permettant de respecter la vie privée. Dans cette optique, on pourra, par exemple, mesurer le nombre de règles d'association manquantes ou rajoutées par rapport à un résultat sans préservation de la vie privée.
– *Le niveau d'incertitude* avec lequel les données sensibles, qui ont été cachées, peuvent être prédites.
– *La résistance* à différentes techniques de fouille de données. Le principe est que les résultats obtenus avec une technique de préservation de la vie privée soient indépendants de l'algorithme de fouille qui sera a posteriori utilisé.

1.5.4 Respect de la vie privée et fouille de données séquentielles

Dans le cadre plus précis qui nous intéresse, c'est-à-dire celui de la fouille de données séquentielles, nous pouvons citer un certain nombre de travaux intéressants.

Dans [ZCM04], le problème de la recherche collaborative de motifs séquentiels en préservant la sensibilité des données de chaque partie est posé. Plus clairement, plusieurs parties possèdent des bases de données et veulent trouver les motifs séquentiels en

mutualisant les bases mais en ne dévoilant pas aux autres parties les items achetés par leurs clients. Elles autorisent la divulgation des identifiants et des dates de chaque transaction. Leur solution est dans un premier temps de transformer les données en données binaires, en codant la présence ou l'absence d'items. Puis ils codent ces vecteurs binaires grâce à des méthodes cryptographiques. Cette approche est cependant assez limitée. En effet, les auteurs supposent que les différentes bases des différentes parties ne partagent pas la même base d'items. Or, si l'on considère, par exemple, les différents magasins d'une même chaîne qui voudraient mutualiser leurs données, il s'agirait bien dans ce cas de produits en commun.

Dans [KPTT06], le même problème de la distribution des données est posé. Les auteurs proposent une extension de l'algorithme SPAM pour trouver les motifs fréquents tout en préservant le caractère privé des données. Leur méthode permet, quant à elle, de considérer de façon globale les items des différentes bases. Ils utilisent des fonctions de cryptage pour sécuriser les données.

Dans un domaine plus précis, celui du flux de données, KIM ET AL. proposent dans [KPWK08] une méthode permettant d'extraire des motifs fréquents dans des données représentant du trafic sur un réseau (par exemple entre différents sites internet) tout en préservant les données. Ce qui est mis en avant dans cet article est le fait de ne pouvoir relier une adresse IP à un site Web. Leur méthode repose sur l'utilisation d'un modèle à N serveurs de dépôt qui permettent de calculer indépendamment des items fréquents. Chaque serveur reçoit un partie des items encryptés et calcule les items fréquents (sans les décrypter). Il envoie ensuite uniquement ceux-ci à un autre serveur qui lui, connaît la clé de décryptage. En résumé, chaque serveur décrypte uniquement une partie des items, et parmi ceux-ci seulement les fréquents. Ils utilisent aussi des techniques permettant de modifier certaines valeurs avec une probabilité donnée.

Enfin, toujours sur les flux de données, GIDOFALVI ET AL. proposent dans [GHP07] une méthode pour trouver des routes fréquemment utilisées, ou des régions spatio-temporelles denses, dans une base de trajectoires tout en préservant l'intimité des individus concernés (il ne faut, par exemple, pas pouvoir retrouver le lieu de travail ou d'habitation des automobilistes). Ils utilisent pour cela des grilles qui permettent de rendre anonyme les données indiquant des positions.

1.5.5 Conclusion

La fouille de données préservant la vie privée est un domaine très délicat à traiter. Il est possible de proposer des outils qui vont permettre de protéger certains types de données, mais il paraît très difficile de trouver une méthode universelle adaptable à tout type de problème. De plus, chacun peut avoir un point de vue très différent sur ce qui relève de la vie privée. Certaines personnes peuvent penser que le simple fait d'extraire des règles d'associations qui proviennent de données les concernant porte atteinte à leur intimité. Il est, malgré tout, important de prendre en considération ce problème et de proposer des solutions aux industriels désireux de faire de la fouille de données dans un contexte où le respect de la vie privée est un facteur essentiel.

Toutes les approches que nous avons présentées, que ce soit pour la fouille de données

transactionnelles ou séquentielles, nécessitent d'avoir des données individuelles. En effet, l'ensemble, sur lequel la fouille est effectuée, est toujours constitué d'enregistrements liés à un individu ou un objet. Chaque séquence provient d'une observation individuelle, et le travail de préservation de la vie privée vient a posteriori. Donc, quelles que soient les techniques appliquées, il aura été nécessaire d'identifier individuellement chaque personne concernée. Nous verrons, dans la dernière partie de ce manuscrit, que dans certaines applications, le problème de préservation de la vie privée peut en partie être résolu par une nouvelle manière de représenter les données sous forme d'automates probabilistes, qui sont au cœur du domaine de l'inférence grammaticale.

2 Inférence grammaticale

Sommaire

Résumé

Dans ce chapitre, nous présentons l'inférence grammaticale régulière. Nous nous intéressons plus particulièrement aux automates probabilistes (PDFA) et à leur apprentissage. Les PDFAs constitueront, dans la suite de ce manuscrit, l'objet mathématique que nous manipulerons pour effectuer de la fouille de données séquentielles.

2.1 Introduction

Rappelons que l'apprentissage automatique, ou apprentissage artificiel [Mit97, CM02] vise à construire des programmes informatiques afin d'apprendre des propriétés sur des données. L'objectif plus particulier de l'apprentissage automatique supervisé est d'adresser un ensemble de données étiquetées à une machine qui les "apprend" et qui doit ensuite être capable de reconnaître de nouvelles données. L'*inférence grammaticale* est un des sous-domaines de l'apprentissage supervisé, adapté aux données structurées ou semi-structurées (*e.g.* séquences ou arbres). Comme nous l'avons déjà évoqué dans le chapitre précédent, nous nous intéresserons dans ce manuscrit seulement aux données séquentielles. Le problème d'apprentissage est alors de trouver, à partir d'un ensemble de données, un modèle, que l'on appelle une grammaire, censé être à l'origine de la génération de ces données. L'ensemble d'apprentissage est composé d'un ensemble de séquences représentant uniquement les mots du langage (données positives) et parfois d'un ensemble de contre-exemples du langage (données négatives). Lors de l'apparition d'une nouvelle donnée, le but est de savoir si elle appartient au langage ou non.

Les grammaires peuvent être classées en plusieurs catégories, selon une hiérarchie initialement proposée par CHOMSKY [Cho57]. Une grammaire est une définition formelle

de la structure syntaxique d'un langage. Elle est définie par un ensemble de règles de production, un ensemble de symboles terminaux qui correspondent à l'alphabet du langage, un ensemble de symboles non-terminaux qui représentent les règles syntaxiques et enfin un élément non terminal qui est l'axiome de départ. Par exemple, si on veut écrire une grammaire correspondant au langage HTML, l'ensemble des terminaux sera l'ensemble des balises HTML existantes (comme `<p>`, `</p>`, `<table>` ou `</table>`) et on pourra, par exemple, trouver les règles suivantes :

- $T \rightarrow$ `<p>` R `</p>` | R et
- $R \rightarrow T$ | `<table>` R `</table>`,

avec T et R des éléments non-terminaux. Par exemple, la règle R se dérive en la règle T ou en la règle R encadrée des balises `<table>` et `</table>`. Le langage engendré par la grammaire est l'ensemble des mots qui peuvent être dérivés à partir de l'axiome. La hiérarchie de Chomsky définit des classes en fonction du type de règles qui la composent. Nous nous intéressons dans cette partie aux grammaires dites *régulières*. Ce sont des grammaires où les règles sont uniquement du type $T \rightarrow aR$ ou $T \rightarrow a$. Un langage engendré par une grammaire régulière sera donc un langage *régulier*. Il a été démontré dans [AU72] que les langages réguliers peuvent être engendrés par des automates finis.

2.2 Les automates

Nous présentons tout d'abord quelques définitions issues de [DM98] qui vont nous permettre d'introduire les automates.

2.2.1 Les automates finis

Définition 2.1 *Soit Σ un alphabet, on notera u,v,w des séquences (aussi notées chaînes et mots) de Σ^*, c'est-à-dire des séquences formées à partir d'éléments de Σ et de longueurs quelconques. On notera ϵ la chaîne vide et $|w|$ la longueur de la séquence w.*

Notons que cette définition est directement applicable aux séquences exploitées dans le domaine de la fouille de données séquentielles vues dans le chapitre précédent.

Définition 2.2 (Préfixe) *u est un préfixe de v ssi il existe w tel que $v = uw$, où uw est la concaténation de u et de w.*

Définition 2.3 (Langage) *Un langage L est un sous-ensemble quelconque de Σ^*. Les éléments de L sont des séquences.*

Définition 2.4 *Soit $Pr(L) = \{u|\exists w,\ uw \in L\}$ l'ensemble des préfixes du langage L. On notera $L/u = \{w|uw \in L\}$ le quotient droit de L par u. Si $u \in Pr(L)$ alors $L/u \neq \emptyset$.*

Définition 2.5 (Automate fini) *Un automate fini* FA *est un quintuplet $A =< Q,\Sigma,q,q_0,F >$ où :*

- *Q est un ensemble fini d'états (on utilisera généralement la notation S ou T pour caractériser les états) ;*

- Σ *est un alphabet fini ;*
- q*:* $Q \times \Sigma \rightarrow Q$ *est une fonction de transition ;*
- q_0 *est l'état initial ;*
- F*:* $F \subset Q$ *est l'ensemble des états finaux.*

Définition 2.6 (Déterminisme) *Un automate fini est déterministe ssi* $\forall S \in Q$ *et* $\forall a \in \Sigma$*, l'ensemble* $\{T|q(S,a) = T\}$ *contient au plus un élément, c'est-à-dire que depuis un état, avec une lettre donnée, on peut accéder à un unique état d'arrivée. On notera* DFA *un automate fini déterministe et* NDFA *un automate fini non déterministe.*

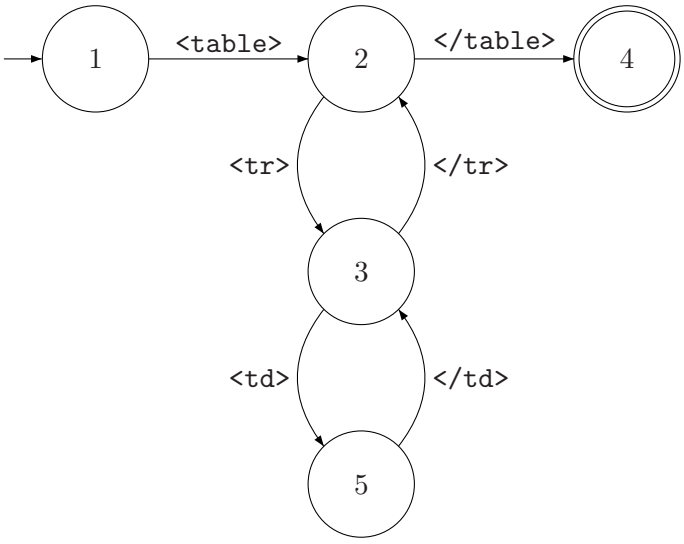

FIG. 2.1 – *Un exemple de* DFA.

L'automate de la figure 2.1 est un DFA, d'alphabet Σ = {`<table>`, `</table>`, `<tr>`, `</tr>`, `<td>`, `</td>` }. L'ensemble des états est $Q = \{1,2,3,4,5\}$, $q_0 = 1$ est l'état initial (caractérisé par une flèche entrante) et les états finaux sont représentés par des doubles cercles (ici $F = \{4\}$). D'autre part, on a, par exemple, $q(2, < tr >) = 3$ qui caractérise la transition entre les états 2 et 3 étiquetée par le symbole `<tr>` .

Définition 2.7 (Acceptation) *L'acceptation d'une chaîne* $u = a_1 a_2 \ldots a_n$ *par un automate* A *déterministe définit une séquence de* $n + 1$ *états* $(S_0,S_1,\ldots,S_n)$ *telle que* $S_0 = q_0$*,* $S_n \in F$ *et* $\forall i \in [0,n-1]$ $q(S_i,a_{i+1}) = S_{i+1}$*. On dit que* S_n *est l'état d'acceptation de la chaîne* u*.*

Par exemple, la séquence (12324) correspond à l'acceptation de la chaîne `<table><tr></tr></table>` dans le DFA de la figure 2.1.

Définition 2.8 *Le langage* $L(A)$ *accepté par un automate* A *est l'ensemble des séquences acceptées par* A*.*

2.2.2 Les automates finis probabilistes

Les automates finis déterministes probabilistes (ou stochastiques) que l'on notera PDFA (pour Probabilistic Deterministic Finite state Automaton) sont une extension des automates finis déterministes (DFA). Le but d'un PDFA est, non seulement de dire si une chaîne appartient au langage défini par l'automate, mais aussi de lui associer une probabilité. Plus formellement, un PDFA peut être défini comme suit :

Définition 2.9 (PDFA**)** *Un automate fini déterministe probabiliste (*PDFA*)* $A =< Q,\Sigma,q,q_0,\pi,\pi_F >$ *est un sextuplet où :*

- Q *est un ensemble fini d'états ;*
- Σ *est un alphabet fini ;*
- q*:* $Q \times \Sigma \rightarrow Q$ *est une fonction de transition ;*
- q_0 *est l'état initial ;*
- π*:* $Q \times \Sigma \rightarrow [0,1]$ *est une fonction de probabilité sur les transitions ;*
- π_F*:* $Q \rightarrow [0,1]$ *est une fonction de probabilité affectant à chaque état une probabilité d'être final.*

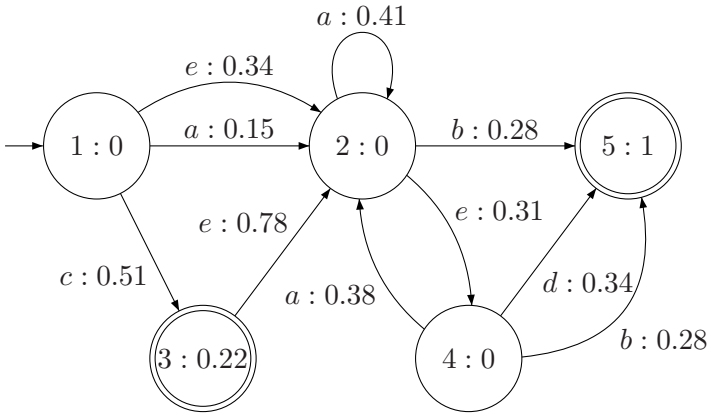

Fig. 2.2 – *Un exemple de* PDFA.

Pour illustrer la définition 2.9, la figure 2.2 montre un exemple de PDFA. Dans cet exemple, $Q = \{1,2,3,4,5\}$ et $\Sigma = \{a,b,c,d,e\}$. L'état initial $q_0 = 1$ est symbolisé par une flèche entrante. Nous représentons sur chaque nœud sa probabilité d'être final. Par exemple $\pi_F(1) = 0$ et $\pi_F(3) = 0.22$. Si l'état a une probabilité non nulle d'être final, il est symbolisé par un double cercle. Sur chaque transition est indiquée la lettre qui lui correspond (par exemple $q(1,a) = 2$), ainsi que la probabilité associée ($\pi(1,a) = 0.15$).

Définition 2.10 (Loi de probabilité) *Une application* $P : \Sigma^* \rightarrow [0,1]$ *définit une loi de probabilité sur* Σ^* *si et seulement si :*

$$\sum_{w \in \Sigma^*} P(w) = 1.$$

Définition 2.11 (Langage stochastique) *Un langage stochastique L est tel que, à chaque mot de L est associée une probabilité et que :*

$$\sum_{w \in L} P(w) = 1.$$

Définition 2.12 (Acceptation) *L'acceptation d'une chaîne* $u = a_1 a_2 \dots a_n$ *par un* PDFA A *définit une séquence de* $n+1$ *états* $(S_0,S_1,\dots,S_n)$ *telle que* $S_0 = q_0$, $\pi_F(S_n) > 0$ *et* $\forall i \in [0,n-1]$ $q(S_i,a_{i+1}) = S_{i+1}$ *et* $\pi(S_i,a_{i+1}) > 0$. *La probabilité de la chaîne u est alors égale à :*

$$P(a_1 a_2 \dots a_n) = \prod_{i=0}^{n-1} (\pi(S_i,a_{i+1})) * \pi_F(S_n).$$

Par exemple, la séquence (122425) correspond à l'acceptation de la chaîne *eaeab* dans l'automate de la figure 2.2. Le calcul de sa probabilité donne :

$$\begin{aligned} P(eaeab) &= \pi(1,e) * \pi(2,a) * \pi(2,e) * \pi(4,a) * \pi(2,b) * \pi_F(5) \\ &= 0.34 * 0.41 * 0.31 * 0.38 * 0.28 * 1 \\ &= 0.0043. \end{aligned}$$

Définition 2.13 (Accessibilité) *Un état* $S \in Q$ *est accessible ssi il existe une chaîne* $u = a_1 a_2 \dots a_n$ *telle qu'il existe une séquence de* $n+1$ *états* $(S_0,S_1,\dots,S_n)$ *avec* $S_0 = q_0$, $\forall i \in [0,n-1]$ $q(S_i,a_{i+1}) = S_{i+1}$ *et* $\pi(S_i,a_{i+1}) > 0$ *et* $S_n = S$.

Concrètement, cela signifie qu'un état est accessible s'il existe un chemin (une suite d'états) pour y accéder depuis l'état initial.

Définition 2.14 (Co-Accessibilité) *Un état* $S \in Q$ *est co-accessible ssi il existe une chaîne* $u = a_1 a_2 \dots a_n$ *telle qu'il existe une séquence de* $n+1$ *états* $(S_0,S_1,\dots,S_n)$ *avec* $S_0 = q$, $\forall i \in [0,n-1]$ $q(S_i,a_{i+1}) = S_{i+1}$ *et* $\pi_F(S_n) > 0$.

Concrètement, cela signifie qu'un état est co-accessible s'il existe un chemin (une suite d'états) pour accéder à un état final depuis cet état.

Définition 2.15 (Utilité) *Un état* $S \in Q$ *est utile ssi il est accessible et co-accessible.*

Dans le PDFA de la figure 2.2 tous les états sont utiles. Nous pouvons énoncer le théorème suivant [Wet80] :

Théorème 2.1 *Un* PDFA $A =< Q,\Sigma,q,q_0,\pi,\pi_F >$ *définit une fonction de probabilité sur* $L(A)$, *donc un langage stochastique ssi :*

$$\forall S \in Q,\ S \text{ est utile et } \sum_{a \in \Sigma \cup \{\#\}} \pi(S,a) = 1,$$

où # *désigne le symbole de terminaison.*

2.3 Inférence grammaticale probabiliste

Comme nous l'avons précisé dans l'introduction, l'inférence grammaticale a pour but de trouver ou d'approcher le langage associé à un ensemble d'exemples. Cet ensemble peut être constitué de mots positifs et négatifs, ou positifs uniquement. Il existe un très grand nombre de résultats théoriques sur l'apprenabilité des différentes classes de langages. En effet, selon la classe des langages [Cho57], et en fonction de l'ensemble de séquences disponible, la complexité algorithmique nécessaire à la découverte du langage ne va pas être la même [Gol78].

D'autre part, selon que l'ensemble d'apprentissage soit composé d'exemples positifs et négatifs, ou positifs seulement, les modèles recherchés seront différents. Dans le premier cas, il va être possible de trouver un modèle exact, c'est-à-dire un DFA qui va reconnaître l'ensemble des données positives et ne pas reconnaître les données négatives. Dans ce cadre, nous pouvons citer différents algorithmes d'apprentissage tels que RPNI [OG92], BLUE-FRINGE [LPP98] ou RPNI* [SJ03] qui est une adaptation de RPNI aux données bruitées. Ces trois algorithmes fonctionnent par fusion d'états. Quand seulement des exemples positifs sont à disposition durant l'apprentissage, les PDFAs sont de bons candidats pour modéliser le langage. On parlera d'*inférence grammaticale probabiliste régulière*. Cette thèse se situe dans ce contexte particulier. Après un rappel sur les cadres possibles d'apprentissage théorique en inférence probabiliste, nous ferons un inventaire des différents algorithmes d'apprentissage de PDFAs.

2.3.1 Cadres théoriques d'apprentissage de PDFAs

Les cadres d'apprentissage que nous présentons ici définissent précisément les critères d'une inférence correcte et les langages qui peuvent être appris selon ces critères. Le premier cadre que nous présentons est *l'apprentissage PAC*. Puis nous aborderons le cadre de *l'identification à la limite*.

Apprentissage PAC

Le cadre de l'apprentissage PAC (probablement approximativement correct) a été introduit par Valiant [Val84]. L'idée sous-jacente est qu'un concept est PAC apprenable s'il peut être approché avec une grande probabilité. Adapté au contexte des PDFAs [KMR+94], le cadre PAC se définit formellement comme suit :

Définition 2.16 (PAC apprenable) *La classe des* PDFA*s est PAC apprenable par un algorithme $\mathcal{A}$ si*

- *pour tout $\epsilon > 0$ et $\delta > 0$*
- *pour toute distribution D sur Σ^* de longueur au plus m*
- *pour tout automate fini probabiliste A à n états (automate cible ayant généré les données)*

$\mathcal{A}$ fournit un automate A', à partir d'un échantillon de taille polynomiale en n, $\frac{1}{\delta}$, $\frac{1}{\epsilon}$ et en temps polynomial en n, $\frac{1}{\delta}$, $\frac{1}{\epsilon}$ et m tel que

$$P(\mathcal{D}_{\mathcal{KL}}(A|A') < \epsilon) > 1 - \delta,$$

où $\mathcal{D}_{\mathcal{KL}}$ est la divergence de Kullback-Leibler.

La divergence de Kullback-Leibler [KL51] permet de mesurer la similarité entre deux automates. Cette mesure utilise les probabilités d'acceptation des chaînes par les automates.

Définition 2.17 (Divergence de Kullback-Leibler) *Étant donnés deux automates A_1 et A_2 définissant deux langages stochastiques L_1 et L_2, la divergence de Kullback-Leibler entre A_1 et A_2 est définie par :*

$$\mathcal{D}_{\mathcal{KL}}(A_1|A_2) = \sum_{w\in\Sigma^*} P_{A_1}(w) * log(\frac{P_{A_1}(w)}{P_{A_2}(w)}).$$

Dans ce cadre strict, il a été démontré que la classe des PDFAs n'était pas PAC-apprenable. En effet, dans [KMR+94], un résultat montre que, pour les fonctions de parité bruitées, le résultat de PAC-apprenabilité est négatif. Il existe par contre des résultats positifs pour des sous-classes de PDFAs. Par exemple, dans [RST98], il est démontré la PAC-apprenabilité des automates stochastiques acycliques. Dans [CT04], Clark et Thollard ont montré que, sous réserve de l'utilisation de paramètres additionnels (borne sur la taille moyenne des chaînes générées, borne sur la taille de l'automate), la classe des PDFAs est apprenable dans le cadre PAC avec la divergence de Kullback-Leibler.

Identification à la limite

A l'inverse du cadre PAC, *l'identification à la limite* définit un cadre d'apprentissage exact. Le but est de retrouver avec exactitude le modèle qui a généré les données. Ce cadre d'apprentissage a tout d'abord été introduit par Gold [Gol78] dans le cadre classique, puis a été étendu dans le cadre stochastique dans [HT00]. Nous donnons une définition dans ce cadre.

Définition 2.18 (Identification à la limite avec probabilité 1) *Une classe de distribution $\mathcal{C}$ sur Σ^* est identifiable à la limite avec probabilité 1 s'il existe un algorithme d'inférence $\mathcal{A}$ qui, pour toute distribution $\mathcal{D} \in \mathcal{C}$, pour toute présentation d'un échantillon de n exemples S_n de $\mathcal{D}$, retourne une distribution $\mathcal{A}(S_n)$ telle que : $\exists i \in \mathbb{N}, \forall k \in \mathbb{N}$, $\mathcal{A}(S_i) = \mathcal{A}(S_k) = \mathcal{D}$.*

Dans [HT00], il est montré que la classe des PDFAs avec une fonction de probabilité à valeurs dans $\mathbb{Q}$ est identifiable à la limite avec probabilité 1.

2.3.2 Cadre général de l'apprentissage des PDFAs

De manière générale un problème d'apprentissage en inférence grammaticale est défini par cinq points :

1. La classe des grammaires à apprendre. Nous traiterons ici uniquement **les grammaires régulières**.

2. L'espace des hypothèses, c'est-à-dire les modèles utilisés pour décrire le langage. Nous aborderons dans le cadre de cette thèse les PDFAs.
3. Le type de présentation des exemples d'apprentissage (voir la section 2.3.3).
4. La classe des méthodes d'inférence (voir la section 2.4).
5. Les critères d'évaluation des hypothèses inférées (voir la section 2.3.4).

2.3.3 Types de présentation des exemples d'apprentissage

Il existe trois types de présentation possible qui peuvent être proposés à l'apprentissage [Gol67]. Bien entendu, ces présentations sont dépendantes de l'application considérée.

- Une *présentation positive* uniquement. C'est une séquence infinie d'éléments du langage L, comportant au moins une présentation de chaque élément de L. Ces éléments sont appelés les exemples positifs du langage.
- Une *présentation négative* uniquement. C'est une séquence infinie d'éléments n'appartenant pas au langage L, comportant au moins une présentation de chaque élément de $\Sigma^* - L$. Ces éléments sont appelés les exemples négatifs ou contre-exemples du langage.
- Une *présentation complète*. C'est une séquence infinie ordonnée de paires (x,d) où $x \in \Sigma^*$ et $d \in \{0,1\}$, comportant au moins une présentation de chaque élément de Σ^*. Pour chaque paire (x,d) si $d = 1$ alors $x \in L$, sinon $x \in \Sigma^* - L$.

Dans le cadre de cette thèse, nous nous placerons dans le même contexte que la fouille de données séquentielles, ce qui implique donc que nous disposerons uniquement d'**une présentation positive**.

2.3.4 Évaluation de l'inférence

Il est nécessaire de disposer d'outils nous permettant de mesurer efficacement la qualité des modèles inférés. En effet, si les cadres théoriques tels que nous les avons présentés ne sont pas applicables en pratique, il doit donc être possible d'estimer dans quelles mesures les modèles appris sont correctement inférés. Les critères utilisés dépendent donc du fait que l'on connaisse ou non l'automate cible.

Mesures entre automates

Dans le premier cas, la divergence de Kullback-Leibler est le candidat souvent retenu. Nous ne revenons pas dessus puisque nous l'avons déjà présentée. Néanmoins, notons que cette mesure a certains inconvénients : elle ne vérifie pas les propriétés d'une distance et elle peut être infinie dans le cas où il existe une chaîne de probabilité nulle dans un des deux automates. Pour ces raisons, différentes autres distances ont été étudiées, comme par exemple la distance d_2 entre deux automates [MdlH04], qui elle, possède toutes les propriétés d'une distance et est toujours calculable.

Dans le cas où l'automate cible n'est pas connu, on fait généralement appel à des mesures de perplexité.

Mesures de perplexité

Définition 2.19 (Perplexité) *La perplexité d'un automate A sur un échantillon LS est égale à :*

$$2^{\frac{1}{||LS||} * \sum_{x \in LS} P_A(x)},$$

où $||LS||$ représente la somme des longueurs des séquences de LS.

Cette mesure permet de connaître l'adéquation entre l'automate inféré et les données d'apprentissage. Plus la valeur est faible et plus le modèle est proche de l'échantillon. Pour appliquer cette mesure, les valeurs des probabilités de chaque séquence doivent être non nulles. Pour cette raison, des techniques de lissage sont appliquées sur la distribution décrite par l'automate. Une solution possible est le lissage par interpolation linéaire [JM80]. Pour cela, on utilise un unigramme (automate universel probabiliste) capable d'affecter une probabilité non nulle à tous les éléments de Σ^*. On le combine ensuite avec l'automate à tester par la formule suivante :

$$P(w) = (1 - \lambda)P_A(w) + \lambda * P_U(w),$$

où $P_U(w)$ est la probabilité de la chaîne dans l'unigramme. Plus la valeur de λ est petite, plus la probabilité affectée par l'automate inféré a un poids important dans le calcul de la perplexité. D'autres méthodes de lissage existent (par exemple la méthode par repli [Kat87]), mais nous ne les présenterons pas ici car elles sont moins courantes et nous ne les avons donc pas utilisées dans les expérimentations réalisées dans cette thèse.

Nous avons présenté précédemment certains cadres théoriques qui définissent des propriétés d'apprenabilité des classes de langages ou d'automates. Nous allons nous concentrer maintenant sur la présentation spécifique des algorithmes respectant le cadre d'apprentissage à la limite, et qui ont été utilisés dans cette thèse.

2.4 Algorithmes d'inférence grammaticale

Nous présentons dans cette section deux principaux algorithmes d'inférence d'automates probabilistes : Alergia [CO94] et Mdi [Tho00]. Nous décrivons le principe commun à ces approches qui fonctionnent **par fusion d'états**.

Afin d'illustrer nos propos, nous introduisons dans le tableau 2.1 un ensemble de 15 séquences qui nous utiliserons par la suite.

ab	bac	baba	abbac	abbaab
ba	abcc	bacc	abccc	baabba
abc	baab	ababc	babac	babaabc

Tab. 2.1 – *Ensemble de 15 séquences construit à partir de l'alphabet $\Sigma = \{a,b,c\}$.*

Les algorithmes que nous présentons sont construits sur le principe suivant. Dans un premier temps, un automate représentant exactement les données présentes dans

l'ensemble d'apprentissage est construit. On appelle cet automate le PPTA (Probabilistic Prefix Tree Acceptor). Ensuite, on généralise les données grâce à un principe de fusion d'états pour s'éloigner d'un automate qui sur-apprendrait les données.

2.4.1 Le PPTA

Définition 2.20 (PPTA) *Le* PPTA $A =< Q,\Sigma,q,q_0,\pi,\pi_F >$, *automate probabiliste accepteur des préfixes d'un ensemble d'apprentissage positif* LS *est un* PDFA *tel que :*

- $Q = Pr(LS)$;
- Σ *est l'alphabet de* LS ;
- $q_0 = \epsilon$;
- $q : \forall S \in Q,\ \forall a \in \Sigma$, *si la chaîne* $S.a \in LS$ $q(S,a) = S.a$;
- $\pi : \forall S \in Q,\ \forall a \in \Sigma,\ \pi(S,a) = \frac{pref(S.a)}{pref(S)}$, *où* $pref(x)$ *est le nombre de chaînes dans* LS *qui ont pour préfixe* x ;
- $\pi_F : \forall S \in Q,\ \pi_F(S) = \frac{occ(S)}{pref(S)}$, *où* $occ(x)$ *désigne le nombre d'occurrences de* x *dans* LS.

Cette définition est basée sur des automates où les états sont nommés par les préfixes qu'ils représentent ; nous n'utiliserons pas cette notation par la suite. En effet, par mesure de lisibilité et pour être plus cohérent avec la présentation des PDFAs que nous avons faite précédemment, nous avons choisi d'étiqueter les états par des chiffres. L'automate de la figure 2.3 représente le PPTA correspondant à l'ensemble d'apprentissage présenté dans le tableau 2.1.

Pour chaque séquence, il existe un chemin unique depuis l'état initial 0, vers un état final. Les séquences de préfixe commun se retrouvent sur la même branche de l'arbre. Chaque transition est caractérisée par un symbole et une probabilité, représentant respectivement la lettre émise et la proportion de séquences venant de l'état précédent et suivant cette transition. Par exemple, la transition entre les états 0 et 1 a une probabilité de $\frac{7}{15}$ associée à la lettre a car, parmi les quinze séquences entrant dans l'état 0, sept commencent par la lettre a et vont à l'état 1. Dans cet exemple, l'état 3 a une probabilité de $\frac{1}{7}$ d'être final, car parmi sept séquences entrantes, une termine dans cet état. Cette automate représente uniquement les données d'origine. Il effectue donc en quelque sorte un apprentissage par cœur des données. Afin d'éviter un phénomène de sur-apprentissage, le PPTA va subir un processus de fusion d'états qui va permettre de généraliser les données.

2.4.2 Le processus de fusion

Nous présentons dans ce qui suit les définitions utiles à l'explication du principe de fusion d'états.

Définition 2.21 (Partition) *Une partition* Π, *sur un ensemble* S, *est un ensemble de sous-ensembles de* S, *non vides, dont les intersections deux à deux sont vides et tels que leur union est égale à* S. *Soit* $s \in S$, $B(s,\Pi)$ *désigne le sous-ensemble, aussi appelé bloc, contenant* s.

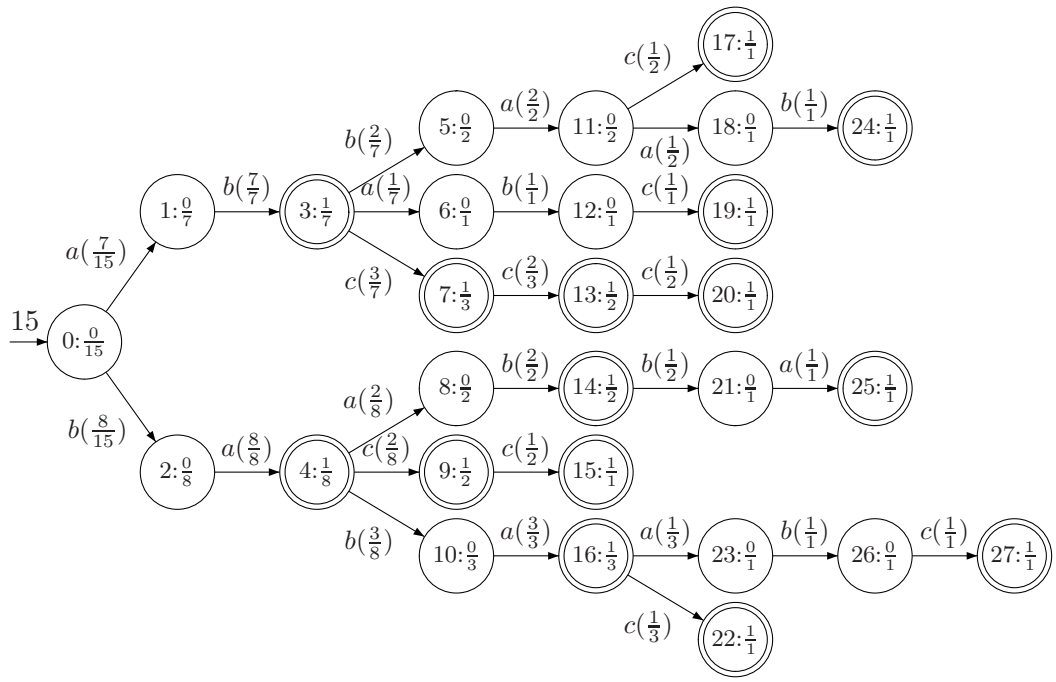

FIG. 2.3 – PPTA *correspondant aux séquences de la table 2.1.*

Définition 2.22 (Automate quotient) *Soit* $\mathcal{A} =< Q_{\mathcal{A}},\Sigma_{\mathcal{A}},q_{\mathcal{A}},q_{\mathcal{A}_0},\pi_{\mathcal{A}},\pi_{\mathcal{A}_F} >$ *un automate probabiliste, soit* Π *une partition de* $Q_{\mathcal{A}}$ *l'ensemble d'états de* $\mathcal{A}$*, l'automate quotient* $\mathcal{B} = \mathcal{A}/\Pi =< Q_{\mathcal{B}},\Sigma_{\mathcal{B}},q_{\mathcal{B}},q_{\mathcal{B}_0},\pi_{\mathcal{B}},\pi_{\mathcal{B}_F} >$ *est défini de la façon suivante :*

- $Q_{\mathcal{B}} = \{B(S,\Pi)|S \in Q_{\mathcal{A}}\}$;
- $\Sigma_{\mathcal{B}} = \Sigma_{\mathcal{A}}$ *(on notera donc simplement* Σ*)* ;
- $q_{\mathcal{B}}$*:* $Q_{\mathcal{B}} \times \Sigma \rightarrow Q_{\mathcal{B}}$, $\forall S,T \in Q_{\mathcal{A}}$, $\forall a \in \Sigma$ *si* $q_{\mathcal{A}}(S,a) = T$ *alors* $q_{\mathcal{B}}(B(S,\Pi),a) = B(T,\Pi)$;
- $B(q_{\mathcal{A}_0},\Pi)$ *est l'état initial ;*
- $\pi_{\mathcal{B}}$*:* $Q_{\mathcal{B}} \times \Sigma \rightarrow [0,1]$, $\forall B \in Q_{\mathcal{B}}$, $\forall a \in \Sigma$ $\pi_{\mathcal{B}}(B,a) = \frac{\sum_{S \in Q_{\mathcal{A}}, S \in B} n_{\mathcal{A}}(S,a)}{\sum_{S \in Q_{\mathcal{A}}, S \in B} n_{\mathcal{A}}(S)}$*, où* $n_{\mathcal{A}}(S,a)$ *représente le nombre d'exemples de LS qui utilisent la transition sortante de S avec la lettre a et* $n_{\mathcal{A}}(S)$ *le nombre d'exemples de LS qui entrent dans l'état S (dans l'automate* $\mathcal{A}$*)* ;
- $\pi_{\mathcal{B}_F}$*:* $Q_{\mathcal{B}} \rightarrow [0,1]$, $\forall B \in Q_{\mathcal{B}}$ $\pi_{\mathcal{B}}(B) = 1 - \sum_{a \in \Sigma}(\frac{\sum_{S \in Q_{\mathcal{A}}, S \in B} n_{\mathcal{A}}(S,a)}{\sum_{S \in Q_{\mathcal{A}}, S \in B} n_{\mathcal{A}}(S)})$.

Reprenons le PPTA de la figure 2.3. Soit la partition $\Pi = \{\{0\},\{1\},\{2\},\{3,4\},\{5,10\},\{6,8\},\{7,9\},\{11,16\},\{12,14\},\{13,15\},\{22,17\},\{18,23\},\{19\},\{20\},\{21\},\{24\},\{26\},\{25\}\}$. L'automate quotient est présenté à la figure 2.4.

Nous pouvons constater que les états $Q_{\mathcal{B}}$ correspondent aux éléments de la partition. L'alphabet est inchangé, $q_{\mathcal{B}_0} = 0$ car il reste identique dans la partition, et on a par exemple $q_{\mathcal{B}}(\{12,14\},b) = 21$ car $q_{\mathcal{A}}(14,b) = 21$ et $B(14,\Pi) = \{12,14\}$.

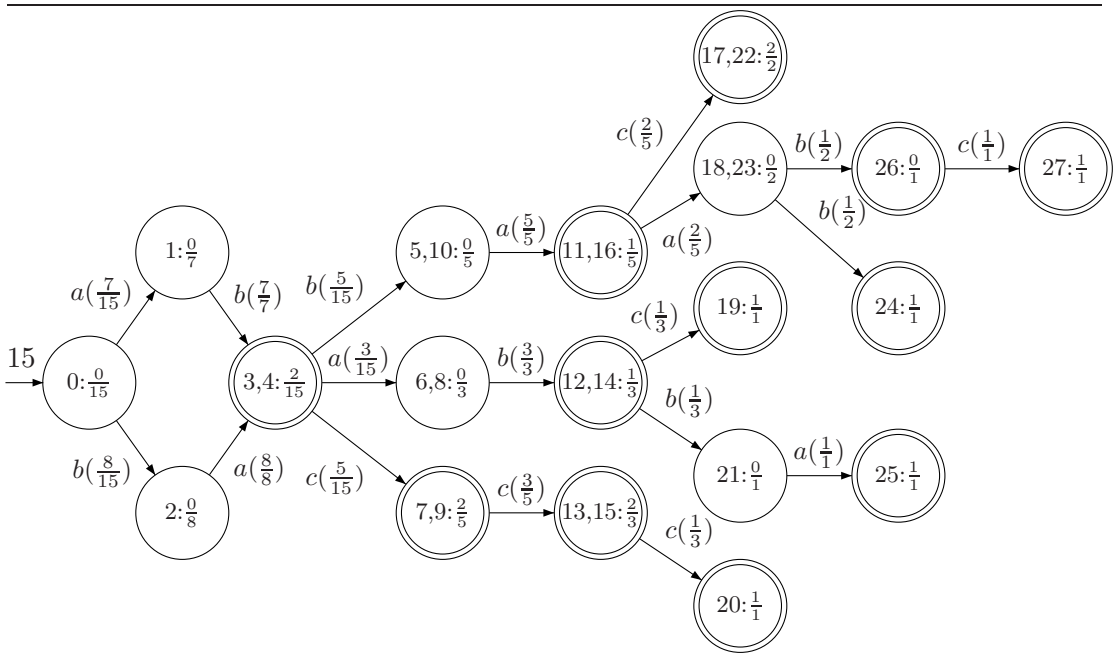

Fig. 2.4 – *Automate quotient non déterministe.*

Pour la fonction de probabilité sur les transitions on obtient :

$$\pi_{\mathcal{B}}(\{5,10\},a) = \frac{n_{\mathcal{A}}(5,a) + n_{\mathcal{A}}(10,a)}{n_{\mathcal{A}}(5) + n_{\mathcal{A}}(10)} = \frac{2+3}{2+3} = \frac{5}{5}.$$

Pour la fonction de probabilité sur les états on obtient :

$$\begin{aligned}
\pi_{\mathcal{B}_F}(\{11,16\}) &= 1 - \left(\frac{n_{\mathcal{A}}(11,a) + n_{\mathcal{A}}(16,a)}{n_{\mathcal{A}}(11) + n_{\mathcal{A}}(16)} + \frac{n_{\mathcal{A}}(11,b) + C_{\mathcal{A}}(16,b)}{n_{\mathcal{A}}(11) + n_{\mathcal{A}}(16)} + \frac{n_{\mathcal{A}}(11,c) + n_{\mathcal{A}}(16,c)}{n_{\mathcal{A}}(11) + n_{\mathcal{A}}(16)}\right) \\
&= 1 - \left(\frac{1+1}{2+3} + \frac{0+0}{2+3} + \frac{1+1}{2+3}\right) \\
&= 1 - \frac{4}{5} = \frac{1}{5}.
\end{aligned}$$

Nous pouvons constater que cet automate n'est pas déterministe. En effet, $q_{\mathcal{B}}(\{18,23\},b) = 24$ et $q_{\mathcal{B}}(\{18,23\},b) = 26$, or la fonction de transition doit être injective, c'est-à-dire qu'en partant d'un état, avec une lettre donnée, on doit arriver au plus dans un état. Il va donc être nécessaire de déterminiser ce nouvel automate. Pour cela construisons une nouvelle partition qui regroupe les états qui violent le déterminisme. Nous devons donc fusionner les états 24 et 26, ce qui donne le résultat de la figure 2.5. Ce processus est réitéré tant que l'automate n'est pas déterministe.

2.4.3 Algorithmes Alergia et Mdi

Comme nous l'avons déjà précisé, les algorithmes d'inférence que nous présentons ici sont basés sur le même principe de fusion comme précédemment décrit. Nous donnons

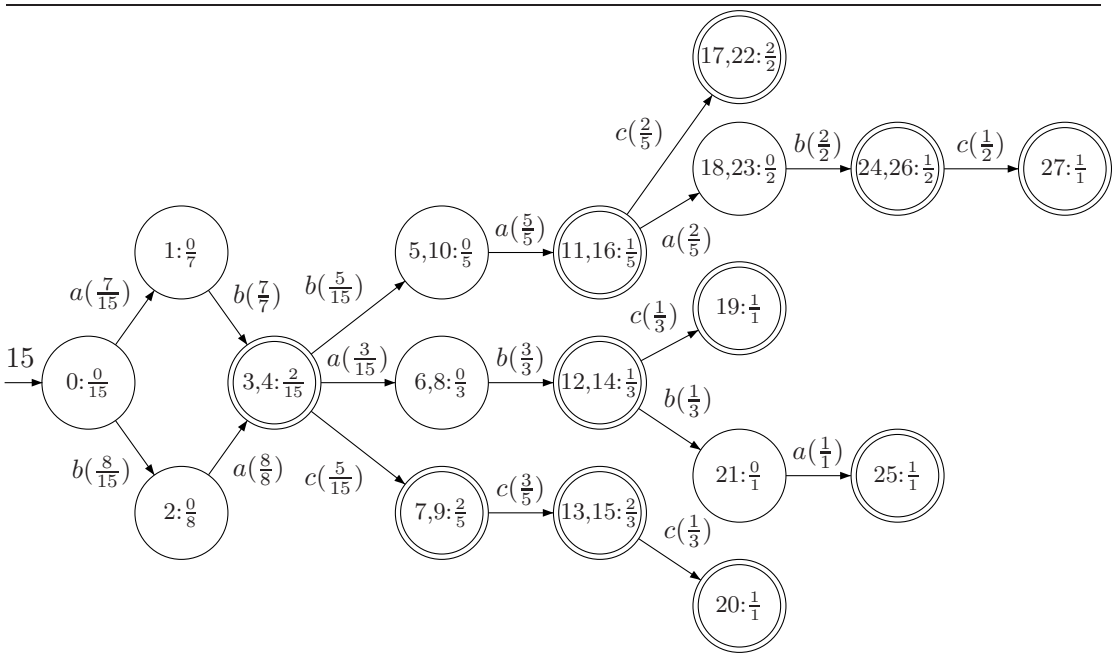

FIG. 2.5 – *Automate quotient après déterminisation.*

tout d'abord un algorithme générique (voir l'algorithme 2.1) décrivant cette catégorie d'approches. Nous détaillerons ensuite deux algorithmes spécifiques : ALERGIA [CO94] et MDI [Tho00].

Algorithme 2.1 : Algorithme générique d'inférence d'automate.

Entrées : Un ensemble d'apprentissage LS
Sorties : Un PDFA
début
  $A \leftarrow$ `Construire_PPTA(`LS`)` ;
  tant que $(S_i,S_j) \leftarrow$ *`Choisir_états(`A`)`* **faire**
    si *`Condition(`S_i, S_j, A`)` vérifiée* **alors**
      $\Pi =$ `Créer_partition(`Q_A,S_i,S_j`)` ;
      $A \leftarrow A/\Pi$;
      $A \leftarrow$ `Déterminiser(`A`)` ;
  retourner A;
fin

La première étape de l'algorithme est la construction du PPTA (fonction `Construire_PPTA`). Ensuite, on choisit parmi les états de l'automate courant (fonction `Choisir_états`) les états qui vont être candidats à la fusion. Cette fonction permet, entre autres, de parcourir l'automate en largeur ou en profondeur d'abord. Nous verrons qu'elle constitue ainsi une première distinction entre les différents algorithmes. Ensuite, on vérifie une certaine condition (fonction `Condition`) qui permet de savoir

si les états choisis sont compatibles avant de les fusionner. Cette condition constitue la seconde différenciation. Enfin, on crée la partition (fonction `Créer_partition`) qui va permettre la fusion des deux états candidats et on calcule l'automate quotient que l'on déterminisera enfin.

Nous présentons maintenant plus en détails deux algorithmes d'inférence de PDFAs que sont Alergia et Mdi. Nous détaillons les spécificités des fonctions de choix des candidats et de condition d'acceptation d'une fusion.

Alergia

Parmi les algorithmes d'inférence grammaticale capables d'apprendre des PDFAs, Alergia [CO94] est probablement le plus connu.

Dans cet algorithme, la fonction de choix des états se fait par ordre hiérarchique. La fonction de compatibilité teste si les états candidats sont suffisamment similaires. Pour ce faire, cette fonction vérifie si les fréquences de chaque symbole sortant des deux états ne sont pas significativement différentes.

Basé sur la borne de Hoeffding [Hoe63], ce test décide que les deux états q_1 et q_2 peuvent être fusionnés *ssi* :

$$\forall z \in \Sigma \cup \{\#\} \quad |\pi(q_1,z) - \pi(q_2,z)| < \sqrt{\frac{1}{2} ln(\frac{2}{\alpha})} \times \left(\frac{1}{\sqrt{n(q_1)}} + \frac{1}{\sqrt{n(q_2)}} \right), \qquad (2.1)$$

où α est un paramètre de généralisation, $n(q_1)$ et $n(q_2)$ sont respectivement le nombre de séquences entrantes dans q_1 et q_2, et $\#$ est le symbole de terminaison d'une séquence. La borne de Hoeffding est aussi vérifiée pour tous les successeurs des états q_1 et q_2.

Expliquons en détails ce processus de fusion à partir du PPTA de la figure 2.3. Considérons la fusion des états 0 et 3, et supposons que nous appliquons le test de Hoeffding avec $\alpha = 0.8$. Premièrement, la borne de Hoeffding est calculée :

$$\begin{aligned} \sqrt{\frac{1}{2} ln(\frac{2}{\alpha})} \times \left(\frac{1}{\sqrt{n(0)}} + \frac{1}{\sqrt{n(3)}} \right) &= \sqrt{\frac{1}{2} ln(\frac{2}{0.8})} \times \left(\frac{1}{\sqrt{15}} + \frac{1}{\sqrt{7}} \right) \\ &= 0.43. \end{aligned}$$

Ensuite, pour chaque lettre $z \in \Sigma \cup \{\#\}$, nous calculons la différence entre les fréquences $|\pi(0,z) - \pi(3,z)|$. Nous obtenons :

- Pour $z = a$: $|\pi(0,a) - \pi(3,a)| = \left|\frac{7}{15} - \frac{1}{7}\right| = 0.32 < 0.43$.
- Pour $z = b$: $|\pi(0,b) - \pi(3,b)| = \left|\frac{8}{15} - \frac{2}{7}\right| = 0.25 < 0.43$.
- Pour $z = c$: $|\pi(0,c) - \pi(3,c)| = \left|\frac{0}{15} - \frac{3}{7}\right| = 0.42 < 0.43$.
- Pour $z = \#$: $|\pi(0,\#) - \pi(3,\#)| = \left|\frac{0}{15} - \frac{1}{7}\right| = 0.14 < 0.43$.

La condition est vérifiée pour toutes les lettres. Supposons que les tests soient aussi vrais pour les successeurs de 0 et 3, les états 0 et 3 peuvent alors être fusionnés, en accord avec la contrainte de compatibilité (voir l'équation 2.1). La figure 2.6 montre l'automate résultat (pour alléger la figure nous ne mentionnons pas les probabilités de transition).

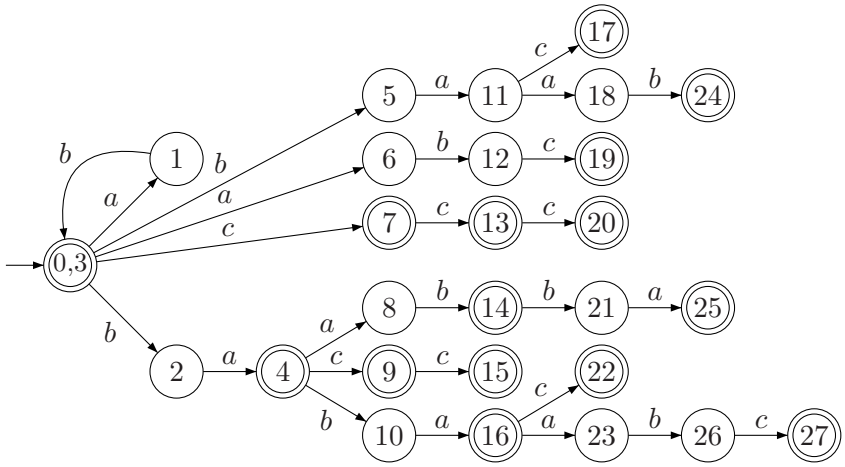

FIG. 2.6 – *Fusion de l'état 0 avec l'état 3.*

Nous pouvons noter que cet automate n'est pas déterministe car il y a deux transitions sortantes de l'état 0 étiquetées avec la lettre b ($q(0,b) = 5$ et $q(0,b) = 2$) et deux transitions sortantes de l'état 0 étiquetées avec la lettre a ($q(0,a) = 1$ et $q(0,a) = 6$). Nous avons donc besoin de fusionner les états d'arrivées 2 avec 5 et 1 avec 6. Créons la partition $\Pi = \{\{0\},\{1,6\},\{2,5\},\{4\},\{7\},\{8\},\{9\},\dots,\}$. Si l'automate résultat n'est toujours pas déterministe, le processus de fusion continue récursivement.

Après une déterminisation complète, la fusion des états 0 et 3 donne l'automate de la figure 2.7 (où les états ont été renommés).

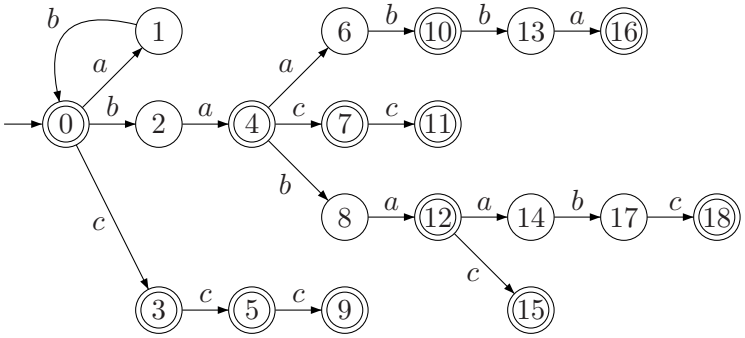

FIG. 2.7 – *Résultat du processus de fusion des états 0 et 3 après plusieurs fusions récursives et un renommage des états.*

En appliquant la définition de l'automate quotient, nous calculons toutes les probabilités de l'automate de la figure 2.7 pour obtenir le PDFA de la figure 2.8.

ALERGIA termine quand plus aucune fusion n'est possible en accord avec la fonction

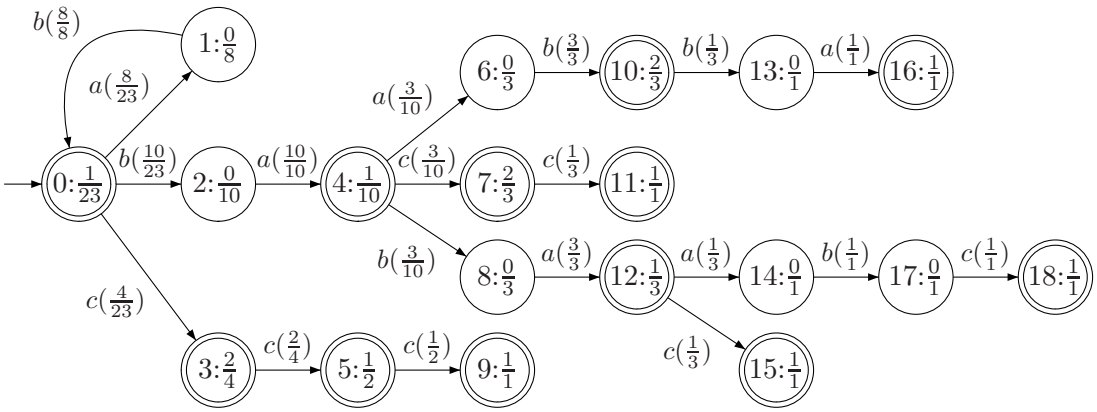

Fig. 2.8 – *Résultat du processus de fusion des états 0 et 3.*

de compatibilité. La figure 2.9 montre le PDFA résultant final. Nous pouvons remarquer que toutes les séquences de la table 2.1 sont modélisées par ce PDFA, c'est-à-dire que leurs chemins d'acceptations terminent dans des états finaux. De plus, grâce aux fusions, nous pouvons noter que ce PDFA généralise les données. Autrement dit, cela signifie qu'il peut représenter d'autres séquences qui n'étaient pas au départ dans l'ensemble d'apprentissage (par exemple, la séquence "ccab" est aussi modélisée). Notons enfin, et ceci sera un paramètre important pour l'utilisation des PDFAs en fouille de données séquentielles, qu'un PDFA constitue une représentation compacte des données.

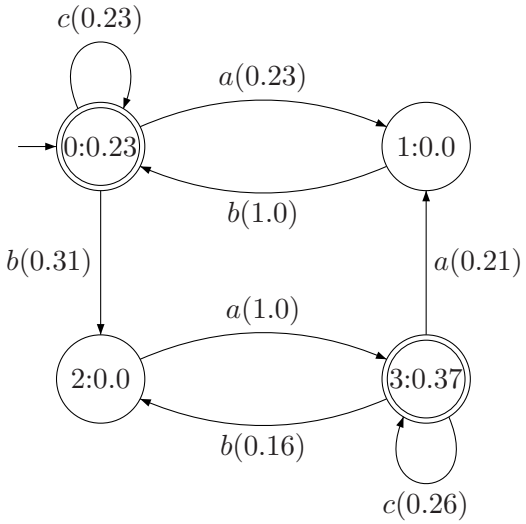

Fig. 2.9 – *Le* PDFA *final, inferré avec* Alergia *à partir des séquences de la table 2.1.*

Il faut mentionner que l'algorithme Alergia a parfois des comportement inadaptés dans des configurations spécifiques, par exemple lorsque les effectifs sur les transitions

sont petits. Plusieurs variantes d'Alergia ont donc été proposées pour résoudre ces problèmes. Par exemple, dans [KD02] Kermorvant et Dupont proposent de remplacer le test de Hoeffding par un test du Khi-deux sur les distributions multinomiales sortantes des états candidats. Ils utilisent aussi une exploration des candidats différente de celle d'Alergia.

Mdi

Dans l'algorithme Alergia, le test de compatibilité des états (test de Hoeffding), bien qu'il soit récursif, est un test local sur les distributions sortantes des états. Dans [Tho00], Thollard propose un nouvel algorithme nommé Mdi qui contourne ce problème. Le but de Mdi est de maximiser, à chaque fusion, la vraisemblance avec l'ensemble des données d'apprentissage. Il propose une heuristique permettant un compromis entre la taille de l'automate que l'on veut le plus petit possible et la divergence avec les données. Il utilise pour cela la divergence de Kullback-Leibler déjà présentée à la définition 2.17.

Le critère de compatibilité utilisé dans Mdi est décrit dans l'algorithme 2.2. Ce critère tient compte de la divergence entre les automates avant et après fusion et du gain de taille en terme de nombre d'états.

Algorithme 2.2 : Fonction de compatibilité de Mdi.

Entrées : Un automate A, deux états ou blocs d'états S_i et S_j, α_M un critère de précision
Sorties : Booléen
début
  $\Pi =$ `Créer_partition(`Q_A,S_i,S_j`)` ;
  $B \leftarrow$ `Déterminiser(`A/Π`)` ;
  retourner $\frac{\mathcal{D}_{\mathcal{KL}}(A,B)}{|Q_A|-|Q_B|} < \alpha_M$
fin

Si le critère n'est pas respecté, la fusion ne sera pas effectuée.

2.5 Conclusion

Nous avons présenté dans ce chapitre l'inférence grammaticale, et plus particulièrement la partie concernant l'apprentissage de pdfas. Ces automates probabilistes présentent trois particularités. Premièrement, ils sont **théoriquement fondés** et peuvent être appris par des algorithmes satisfaisant les conditions du cadre d'apprenabilité à la limite. Deuxièmement, **ils couvrent non seulement les séquences d'apprentissage mais généralisent celles-ci** par un processus de fusion d'états. Enfin, ils constituent une **représentation condensée** des séquences d'origine.

Nous pensons donc qu'ils constituent un objet extrêmement intéressant à manipuler pour effectuer de la fouille de données séquentielles, sous réserve de détenir les outils

mathématiques pour en extraire les sous-séquences fréquentes. Les premiers travaux pionniers exploitant les PDFAs dans ce domaine sont ceux d'Hingston [Hin02], que nous présentons dans le chapitre suivant, et que nous étendrons dans la suite de ce manuscrit.

3 Fouille d'Automates Probabilistes

Sommaire

Résumé

Dans ce chapitre, nous présentons l'approche proposée par HINGSTON [Hin02] qui consiste à exploiter les PDFAs appris par des techniques d'inférence grammaticale pour faire de la fouille de séquences. Nous présentons en détails cette méthode (ainsi que ses limites) car elle servira de point de départ de nos contributions dans la suite de ce manuscrit.

3.1 Introduction

Nous avons présenté dans le premier chapitre le cadre de la fouille de données séquentielles, dont le but est d'extraire à partir d'un ensemble d'exemples des sous-séquences fréquentes. Nous présentons dans ce chapitre la première approche proposée dans la littérature [Hin02] visant à faire de la fouille de séquences à partir d'automates. L'idée est d'inférer un PDFA à partir des séquences à fouiller, grâce à un algorithme type ALERGIA et ensuite d'utiliser cet automate pour calculer la probabilité des sous-séquences candidates.

Dans son approche, HINGSTON justifie son utilisation des PDFAs par le fait que ces derniers constituent une représentation compacte des données, ce qui peut être un réel avantage lorsque la quantité de données à traiter est très importante. En effet, comme nous avons pu le souligner dans le premier chapitre, la quantité des données exploitables par les algorithmes de fouille de données devient de plus en plus grande. La recherche de modèles plus compacts est donc un enjeu important. Néanmoins, l'extraction des motifs (notamment avec des trous) à partir d'un PDFA doit être rendue possible par la mise en place d'outils mathématiques efficaces. HINGSTON présente ainsi une technique permettant de calculer la fréquence de n'importe quel motif à partir d'un PDFA. Nous

proposons dans la section suivante de rentrer dans les détails mathématiques de cette approche que nous serons amenés à généraliser dans la suite du manuscrit.

3.2 Formalisme de l'extraction de motifs à partir de PDFA

Supposons que nous disposons de l'ensemble de séquences déjà utilisé au chapitre précédent, et que nous rappelons dans le tableau 3.1. Dans un premier temps, un PDFA $A =< Q,\Sigma,q,q_0,\pi,\pi_F >$ est appris grâce à l'algorithme Alergia (voir le chapitre 2). Le PDFA obtenu est rappelé à la figure 3.1. Comment extraire de A la probabilité de n'importe quel motif?

ab	bac	baba	abbac	abbaab
ba	abcc	bacc	abccc	baabba
abc	baab	ababc	babac	babaabc

Tab. 3.1 – *Ensemble de 15 séquences construit à partir de l'alphabet* $\Sigma = \{a,b,c\}$.

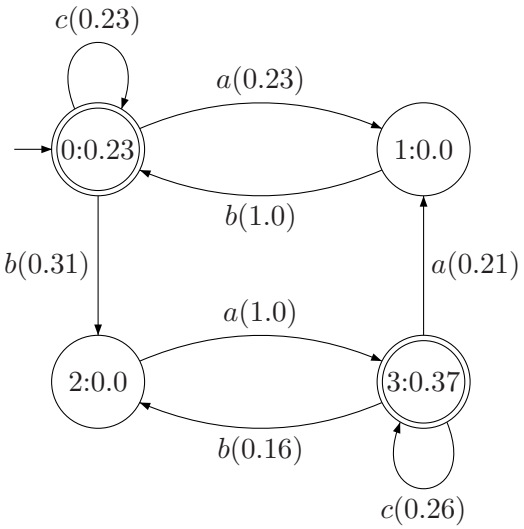

Fig. 3.1 – PDFA *inféré par* Alergia *depuis les séquences de la table 3.1.*

3.2.1 Notations

Introduisons les notations que nous allons utiliser :

- $x \in \Sigma$ décrit un symbole unique.
- $w =< x_1x_2 \dots x_l >$ décrit un motif contenant l symboles **potentiellement non consécutifs**.
- $P(S,w)$ décrit la probabilité d'obtenir w à partir de l'état S de A.

– $P(w)$ décrit la probabilité de w **estimée** à partir de l'automate, ce qui correspond à $P(q_0,w)$ la probabilité d'obtenir w à partir de l'état initial q_0 de A.
– $p(w)$ décrit la probabilité **réelle** dans la base de séquences (celle-ci sera estimée par $P(w)$).

Rappelons que nous avons déjà présenté dans le chapitre 2.2.2 le calcul de la probabilité d'acceptation d'une chaîne dans un automate. Mais, une chaîne d'acceptation implique que l'on commence le parcours à l'état initial, que tous les éléments soient consécutifs et que l'on termine dans un état final. Dans le cas d'un sous-motif, les éléments ne sont pas nécessairement consécutifs et l'arrêt dans un état final n'est pas obligatoire. Le calcul de la probabilité d'un tel sous-motif devient donc beaucoup plus complexe.

Étudions d'abord le cas d'un motif à un symbole avant de le généraliser aux sous-motifs de taille quelconque.

3.2.2 Proportion d'un symbole $P(x)$

Les séquences contenant le symbole x peuvent être séparées en deux groupes : les séquences commençant par x et celles ne commençant pas par x. C'est cette idée qu'HINGSTON a exploitée pour calculer $P(S,x)$.

Prenons l'exemple du calcul de la proportion de séquences contenant le symbole c dans l'automate 3.1 ; ceci nécessite donc de calculer $P(0,c)$ (0 étant l'état initial). En observant l'automate, nous déduisons que $P(0,c) = 0.23 + 0.23 * P(1,c) + 0.31 * P(2,c)$. En effet, pour obtenir un c à partir de l'état 0 il y a trois possibilités :

– émettre un c avec une probabilité $\pi(0,c) = 0.23$, ce qui correspond aux séquences commençant par c,
– émettre un a avec une probabilité $\pi(0,a) = 0.23$ puis avoir un c à partir de l'état d'arrivée $q(0,a) = 1$,
– émettre un b avec une probabilité $\pi(0,b) = 0.31$ puis avoir un c à partir de l'état d'arrivée $q(0,b) = 2$.

En adoptant le même principe, on obtient : $P(1,c) = 1 * P(0,c)$, $P(2,c) = 1 * P(3,c)$ et $P(3,c) = 0.26 + 0.21 * P(1,c) + 0.16 * P(2,c)$. Ces équations constituent un système d'équations linéaires que nous devons résoudre pour obtenir $P(0,c)$.

En généralisant ces équations, nous obtenons la formule récursive suivante :

$$P(S,x) = \pi(S,x) + \sum_{z \neq x \in \Sigma} \left(\pi(S,z) \times P(q(S,z),x)\right), \tag{3.1}$$

qui peut être réécrite comme suit :

$$P(S,x) = \pi(S,x) + \sum_{T \in Q} \left(\sum_{z \neq x, q(S,z)=T} \pi(S,z) \right) \times P(T,x). \tag{3.2}$$

Pour résoudre ces systèmes d'équations linéaires, HINGSTON réécrit le problème sous forme matricielle, qui, grâce aux outils d'algèbre linéaires, peut être résolu efficacement.

Soit $\rho(x)$ la matrice de composantes

$$\rho_{S,T}(x) = \sum_{z \neq x, q(S,z)=T} \pi(S,z).$$

$\rho_{S,T}(x)$ décrit simplement la probabilité d'utiliser une transition différente de x entre les états S et T. Soit $P(x)$ (resp. $\pi(x)$) les vecteurs des valeurs de $P(S,x)$ (resp. $\pi(S,x)$), l'équation 3.2 devient :

$$P(x) = \pi(x) + \rho(x) \times P(x),$$

puis

$$P(x) = (I - \rho(x))^{-1} \times \pi(x),$$

où I est la matrice identité. Reprenons notre exemple et calculons $P(q_0,c)$. Le vecteur $\pi(c)$ a les composantes $\pi(0,c) = 0.23$, $\pi(1,c) = 0.0$, $\pi(2,c) = 0.0$, $\pi(3,c) = 0.26$. Nous avons donc

$$\pi(c) = \begin{pmatrix} 0.23 \\ 0 \\ 0 \\ 0.26 \end{pmatrix}.$$

Pour les matrices $\rho(c)$ et $(I - \rho(c))^{-1}$, nous obtenons

$$\rho(c) = \begin{pmatrix} 0 & 0.23 & 0.31 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0.21 & 0.16 & 0 \end{pmatrix}$$

et

$$(I - \rho(c))^{-1} = \begin{pmatrix} 1.44 & 0.44 & 0.53 & 0.53 \\ 1.44 & 1.44 & 0.53 & 0.53 \\ 0.36 & 0.36 & 1.32 & 1.32 \\ 0.36 & 0.36 & 0.32 & 1.32 \end{pmatrix}.$$

Ce qui donne,

$$P(c) = \begin{pmatrix} 1.44 & 0.44 & 0.53 & 0.53 \\ 1.44 & 1.44 & 0.53 & 0.53 \\ 0.36 & 0.36 & 1.32 & 1.32 \\ 0.36 & 0.36 & 0.32 & 1.32 \end{pmatrix} \times \begin{pmatrix} 0.23 \\ 0 \\ 0 \\ 0.26 \end{pmatrix} = \begin{pmatrix} 0.469 \\ 0.469 \\ 0.426 \\ 0.426 \end{pmatrix}.$$

Nous déduisons donc que $P(0,c) = 0.469$. Rappelons que cette valeur est une estimation (puisqu'issue d'un PDFA appris par généralisation) de la proportion réelle des séquences qui, dans la base de données de départ (voir la table 3.1), contiennent effectivement le symbole c. En calculant cette proportion dans la table 3.1 nous obtenons $p(c) = \frac{9}{15} = 0.6$. Cette différence peut paraître importante. Elle est surtout due ici au fait que le PDFA a été appris à partir d'un petit exemple jouet de 15 séquences seulement. Nous reviendrons dans un chapitre suivant sur les conditions nécessaires pour apprendre un "bon" PDFA. Ceci constituera d'ailleurs une des contributions majeures de cette thèse. Mettons pour l'instant ce biais statistique de côté.

3.2.3 Proportion d'un motif avec trou $P(w)$

Basé sur le même principe, nous pouvons estimer la probabilité $p(w)$ d'un motif $w =< x_1...x_l >$. Soit $F(S,T,x_1)$ la probabilité qu'un chemin aléatoire commençant à l'état S et finissant à l'état T contienne exactement un symbole x_1 à la dernière position. HINGSTON utilise un raisonnement similaire à celui du calcul de $P(x)$ pour montrer que :

$$F(S,T,x_1) = \sum_{x_j \neq x_1} (\pi(S,x_j) \times F(q(S,x_j),T,x_1)) + \begin{cases} \pi(S,x_1) & si\ q(S,x_1) = T \\ 0 & sinon. \end{cases} \quad (3.3)$$

L'équation 3.3 peut être réécrite en utilisant la matrice $\gamma(x_1)$ de valeurs

$$\gamma(S,T,x_1) = \begin{cases} \pi(S,x_1) & si\ q(S,x_1) = T \\ 0 & sinon. \end{cases} \quad (3.4)$$

Soit $F(x_1)$ la matrice de valeurs $F(S,T,x_1)$, l'équation 3.3 devient :

$$F(x_1) = \gamma(x_1) + \rho(x_1) \times F(x_1),$$

et comme précédemment, nous déduisons :

$$F(x_1) = (I - \rho(x_1))^{-1} \times \gamma(x_1).$$

Expliquons désormais comment calculer $P(q_0, < x_1...x_l >)$, l'estimation de $p(< x_1...x_l >)$. Focalisons-nous dans un premier temps sur le cas $l = 2$, c'est-à-dire $P(S, < x_1x_2 >)$. Notons qu'une séquence contenant x_1 suivi par x_2 peut être divisée en une partie contenant le premier x_1 apparaissant dans la séquence, et une seconde partie contenant x_2. Nous pouvons en déduire que :

$$P(S, < x_1x_2 >) = \sum_T F(S,T,x_1) \times P(T,x_2). \quad (3.5)$$

En utilisant la forme matricielle, nous obtenons

$$P(< x_1x_2 >) = F(x_1) \times P(x_2).$$

En généralisant ce principe à l symboles, nous avons

$$P(< x_1...x_l >) = F(x_1) \times ... \times F(x_{l-1}) \times P(x_l). \quad (3.6)$$

Par exemple, avec le PDFA de la figure 3.1, nous pouvons estimer la probabilité $p(< cc >)$ de séquences contenant le motif $< cc >$. Nous devons donc calculer $P(0, < cc >)$. Nous avons besoin de $\gamma(c)$:

$$\gamma(c) = \begin{pmatrix} 0.23 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0. & 0 & 0 & 0.26 \end{pmatrix},$$

pour calculer $F(c)$:

$$F(c) = \begin{pmatrix} 1.44 & 0.44 & 0.53 & 0.53 \\ 1.44 & 1.44 & 0.53 & 0.53 \\ 0.36 & 0.36 & 1.32 & 1.32 \\ 0.36 & 0.36 & 0.32 & 1.32 \end{pmatrix} \times \begin{pmatrix} 0.23 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.26 \end{pmatrix} = \begin{pmatrix} 0.33 & 0 & 0 & 0.14 \\ 0.33 & 0 & 0 & 0.14 \\ 0.08 & 0 & 0 & 0.34 \\ 0.08 & 0 & 0 & 0.34 \end{pmatrix}.$$

Donc

$$P(< cc >) = \begin{pmatrix} 0.33 & 0 & 0 & 0.14 \\ 0.33 & 0 & 0 & 0.14 \\ 0.08 & 0 & 0 & 0.34 \\ 0.08 & 0 & 0 & 0.34 \end{pmatrix} \times \begin{pmatrix} 0.469 \\ 0.469 \\ 0.426 \\ 0.426 \end{pmatrix} = \begin{pmatrix} 0.21 \\ 0.21 \\ 0.18 \\ 0.18 \end{pmatrix}.$$

Nous en déduisons que $P(0, < cc >) = 0.21$. Comme pour $p(c)$, nous pouvons observer dans le tableau 3.1 que la vraie probabilité $p(< cc >) = \frac{3}{15} = 0.2$.

3.3 Autres travaux connexes

A notre connaissance, il n'existe pas de travaux similaires sur l'utilisation d'un automate stochastique qui permette d'extraire des motifs séquentiels.

Cependant, nous pouvons citer les travaux, plus ou moins connexes, de Borges et Levene ([BL98], [BL99], [BL04], [BL05] et [BL07]) qui ont proposé une approche pour traiter des données telles que les sessions de navigation des utilisateurs sur des sites Web. Leurs méthodes visent à généraliser les séquences de navigation à l'aide de modèles probabilistes, mais dans le cas de séquences de lettres contiguës. Leur but est d'améliorer la qualité des services proposés par un site Web en utilisant divers critères tels que la prédiction de la prochaine page visitée. Ils ont choisi d'utiliser des modèles de Markov, à partir desquels ils extraient des probabilités de parcours. Les modèles sont construits, à partir des sessions, de telle façon que chaque page corresponde à un état du modèle de Markov et chaque couple représentant des pages visitées successivement soit codé par une transition. La probabilité d'une transition correspond au nombre de fois où cette dernière est traversée divisé par le nombre de fois où l'état de départ est visité. Dans un premier temps, ils ont utilisé des modèles de Markov de premier ordre (dans [BL99]) puis, du fait de la faible précision de ces modèles, ils ont étendu leur technique à des modèles d'ordre supérieur. Dans les chaînes de Markov d'ordre supérieur, les états représentent des ensembles de pages, ce qui augmente grandement la taille des modèles mais aussi leur exactitude par rapport aux données. Par exemple, dans un modèle d'ordre 2, un état correspond à deux pages consécutives rencontrées. Les modèles d'ordre supérieur à 1 permettent ainsi de capturer des dépendances à plus long terme. Toujours avec le même objectif, ils ont proposé un modèle utilisant des VLMC (chaînes de Markov à longueur variable) qui permettent d'avoir une taille de l'historique variable. Ces modèles sont construits par un processus de clonage d'états pour différencier les différents chemins menant à un même état. Si l'on voulait faire le parallèle avec les automates pour des modèles d'ordre 2 cela signifierait que, partant

du ppta construit sur les sessions, il ne faut pas fusionner deux états si les chemins de longueur 2 menant à ces états ne sont pas identiques. Plus précisément, soit S_1 et S_2 les deux états considérés, si $S_1^{'}$ et $S_2^{'}$ les antécédents de S_1 et S_2 sont tels que $q(q(S_1^{'},a),b) = S_1$ et $q(q(S_2^{'},a),b) = S_2$ avec a et b quelconque appartenant à Σ alors les états peuvent être fusionnés. Cette technique permet de garder l'historique des chemins, ici l'historique est de longueur 2. Le modèle de Markov d'ordre maximal correspondrait, dans sa structure, au ppta. Le clonage n'est pas effectué pour tous les états, mais uniquement lorsque la différence entre les probabilités d'un modèle et de l'ordre supérieur est suffisamment grande, c'est-à-dire si le gain apporté par le modèle supérieur est suffisamment significatif. En effet, plus le modèle est gros plus les parcours et les calculs sont coûteux. Ils proposent aussi une mesure de "résumé" qui permet de mesurer si le modèle est en adéquation avec les données. Leurs méthodes donnent de bons résultats pour prédire par exemple la dernière page d'une session. Cependant, Borges et Levene ne proposent pas de méthodes permettant de calculer les probabilité des motifs constitués de pages non consécutives.

On peut enfin citer les travaux de Callut et Dupont ([Cal07] et [DCD+06]), qui extraient des sous-graphes pertinents entre des nœuds d'intérêts d'un graphe, grâce à des hmms. Le but est de capturer les relations entre ces nœuds. Après avoir effectué des transformations sur le graphe pour obtenir des hmms, ils utilisent des marches aléatoires pour calculer ces sous-graphes, en considérant les fréquences d'utilisation de chacune des arêtes. Ici encore, les motifs extraits sont des composantes connexes.

3.4 Limites et perspectives de l'approche d'Hingston

La technique proposée par Hingston pour approximer, à partir d'un modèle probabiliste, la probabilité d'un motif est tout à fait originale. Elle dispose des avantages suivant :

1. Elle propose d'exploiter une **représentation condensée** (un pdfa) des séquences d'origine.
2. Par le principe de généralisation des séquences, elle ouvre la porte à **l'extraction de nouveaux motifs**, non présents dans les données d'origine.
3. Elle propose un système original d'interrogation, par requêtes sur le pdfa, **en calculant efficacement la probabilité** de n'importe quel type de motif non consécutif.

Néanmoins, l'approche d'Hingston présente un certain nombre de limites et de perspectives non encore abordées :

1. Aucun résultat théorique n'est fourni sur le nombre minimal de séquences nécessaires à la construction d'un "bon" pdfa. Ou, plus généralement, combien de séquences sont nécessaires pour faire de la fouille de données séquentielles pertinente ?
2. Comme nous l'avons vu au chapitre 1, la tendance actuelle en fouille de données est d'intégrer des contraintes dans les systèmes d'extraction afin de pouvoir passer à l'échelle et gérer des bases de séquences de plus en plus grandes. Proposer et

gérer de telles contraintes dans le cadre de la fouille de PDFAs est un problème ouvert qui n'a pas été abordé par Hingston.

3. Enfin, exploiter un PDFA plutôt que les séquences elles-mêmes semble être une stratégie qui permettrait de résoudre des problèmes de préservation de vie privée. Ce point de vue là n'a pas été encore abordé.

L'objectif de la prochaine partie est de présenter les principales contributions apportées dans cette thèse et qui visent à aborder chacun des trois points précédemment cités.

Deuxième partie

Apports de l'inférence grammaticale pour la fouille de données séquentielles

4 Contributions à la fouille de données séquentielles dans un cadre statistique

Sommaire

Résumé

Nous présentons dans ce chapitre les avantages qu'apporte la fouille de données séquentielles dans un cadre probabiliste. Nous apportons trois contributions majeures dans ce contexte de travail original. La première consiste à fournir, en utilisant les risques de type I et II de tests statistiques, une borne sur le nombre de séquences nécessaires pour assurer un processus de fouille pertinent. La deuxième vise à s'assurer de la construction de "bons" PDFAs pour extraire des estimateurs corrects des vraies probabilités d'apparition des motifs. Enfin, pour faire face au très grand nombre potentiel de motifs à extraire, nous proposons des contraintes probabilistes applicables à la fouille de PDFAs.

4.1 Introduction

Comme nous l'avons vu au chapitre 1, beaucoup d'algorithmes de fouille de données séquentielles ont été proposés au cours des dernières années [MTV97, Zak00, GRS02, PHW02, KPWD03, YHA03, CWC04]. Pour la plupart d'entre eux, le but est d'améliorer la complexité, c'est-à-dire l'efficacité en temps de calcul et en espace mémoire utilisé pour le processus de fouille. Les critères de qualité de ces algorithmes reposent principalement sur le fait qu'ils soient corrects et complets, c'est-à-dire que tous les motifs fréquents soient extraits (*complétude*) et uniquement ceux-ci (*correction*).

D'un point de vue statistique, on peut noter qu'un processus de fouille de données séquentielles est presque toujours effectué sur un ensemble fini d'exemples (LS) qui a été construit à partir d'une distribution cible, généralement inconnue. Par exemple, LS peut être un ensemble de phrases issues de la langue française qui est par nature de dimension infinie. Il est aussi important de noter que dans certains domaines, il peut être difficile de collecter de larges quantités de données. L'acquisition des séquences peut en effet être extrêmement coûteuse, en temps ou en argent. C'est très souvent le cas lorsque l'on considère, par exemple, des données médicales ou biologiques, dont la collecte nécessite parfois des processus expérimentaux lourds. On peut aussi être confronté à des cas réels où, tout simplement, le nombre d'évènements observés est très faible (les krachs boursiers, par exemple).

Malgré tout ce qui vient d'être évoqué, les algorithmes de fouille de données séquentielles sont presque toujours appliqués sans aucune considération de la distribution statistique sous-jacente, à partir de laquelle LS a été construit. Autrement dit, l'algorithme de fouille n'évalue en aucune façon la vraisemblance qu'un motif extrait soit un artefact de l'échantillonnage plutôt qu'un motif consistant avec la distribution cible. Et évidemment, plus la taille de l'échantillon est petite et plus les risques d'erreurs statistiques seront grands.

Le fait de vérifier que le résultat d'un processus de fouille soit complet et correct par rapport à un échantillon LS ne garantit donc pas que les motifs extraits décrivent réellement une information pertinente pour la distribution statistique sous-jacente. Des motifs peuvent être extraits uniquement par chance, alors que d'autres motifs réellement fréquents peuvent être oubliés, à cause de l'échantillonnage.

Dans la section suivante, nous allons prendre en compte ce problème d'un point de vue théorique, pour vérifier sous quelles contraintes un processus de fouille de données séquentielles est statistiquement pertinent. Nous montrons que décider si un motif est fréquent ou pas peut être assimilé à un test statistique de proportion. Nous présentons alors une borne sur le nombre de séquences permettant de contrôler les erreurs de types I et II. Nous illustrons l'utilisation de cette borne par différentes expérimentations. Nous montrons ensuite que si cette borne n'est pas vérifiée, il devient alors encore plus intéressant, comme nous l'avons vu au chapitre 3, d'exploiter des PDFAs qui généralisent les séquences d'origine, ou en d'autres termes couvrent beaucoup plus de séquences que celles de LS. Nous donnons alors, ce qui constituera notre seconde contribution, les conditions nécessaires pour assurer qu'un PDFA soit utile pour la fouille (voir la section 4.3). Enfin, nous introduisons de nouvelles contraintes probabilistes applicables sur les PDFAs, afin de réduire le nombre de motifs extraits, ce qui constituera notre troisième contribution (voir la section 4.4).

4.2 Erreurs en fouille de données séquentielles et borne théorique

4.2.1 Exemple introductif

En fonction des contraintes appliquées, les sous-séquences extraites d'un ensemble de données séquentielles correspondent exactement aux motifs possédant les caractéristiques requises dans la base de données à traiter. Par exemple, si le support d'une sous-séquence, dans les données LS dont on dispose, vaut 0.39 et que le seuil de support demandé est de 0.4 cette séquence ne sera pas retenue comme sous-séquence fréquente. Considérons un exemple pour illustrer les limites de cette approche. Supposons qu'une pièce de monnaie soit lancée 10 fois de suite ($N = 10$). On obtient une base LS de 10 séquences composées ici chacune d'un seul item (<*pile*> ou <*face*>). Supposons que nous observons, dans LS, respectivement huit <*pile*> et deux <*face*>. Si nous fixons un seuil de support $p_0 = 0.5$, la séquence <*pile*> sera considérée comme fréquente dans LS par n'importe quel algorithme de fouille de données séquentielles, car ils observent une probabilité $\hat{p} = 0.8$ supérieure à p_0. D'un autre coté, la séquence <*face*> ne sera pas considérée comme fréquente. Cela induit-il que la connaissance extraite " <*pile*> *est plus fréquent que* <*face*>" est significative? En fait, il est facilement démontrable qu'une telle configuration de <*pile*> et <*face*> peut se produire souvent si on lance la pièce de monnaie uniquement 10 fois, et ce, sans remettre en cause l'équilibre de la pièce. Si on obtenait les mêmes proportions de <*pile*> et de <*face*> pour 10 000 lancés, les conclusions seraient assurément différentes. Nous pouvons donc noter ici que la taille de la base de données séquentielles à traiter a une grande influence sur le résultat. Par cet exemple simplissime, on constate qu'un processus de fouille de données séquentielles peut aboutir à des conclusions erronées.

4.2.2 Formalisation

Soit LS un échantillon de séquences issu d'une distribution sous-jacente D, potentiellement infinie, que nous représentons par soucis de simplification par l'échantillon D dans la figure 4.1.

Étant donné un motif y (dans les graphiques suivants, y modélise des séquences contenant le motif y et x des séquences ne contenant pas le motif y), et un seuil de support fixé arbitrairement à 60%. Quelles sont les situations possibles pour y?

- y est un vrai positif (noté VP), c'est-à-dire qu'il est fréquent dans LS, mais aussi selon D. Par exemple, dans la figure 4.2, y a un support de 70% dans LS, et de 65% dans D.
- y est un faux positif (noté FP), c'est-à-dire qu'il est fréquent dans LS, mais pas selon D. Par exemple, dans la figure 4.3, y a un support de 60% dans LS, et de 50% dans D.
- y est un faux négatif (noté FN), c'est-à-dire qu'il n'est pas fréquent dans LS mais fréquent selon D. Par exemple, dans la figure 4.4, y a un support de 50% dans LS, et de 65% dans D.

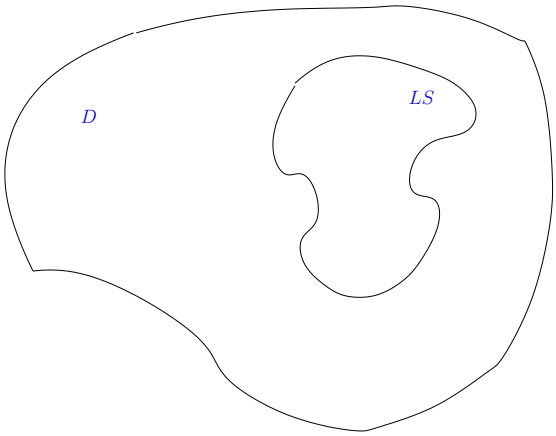

Fig. 4.1 – *Échantillon LS issue d'un distribution sous-jacente D.*

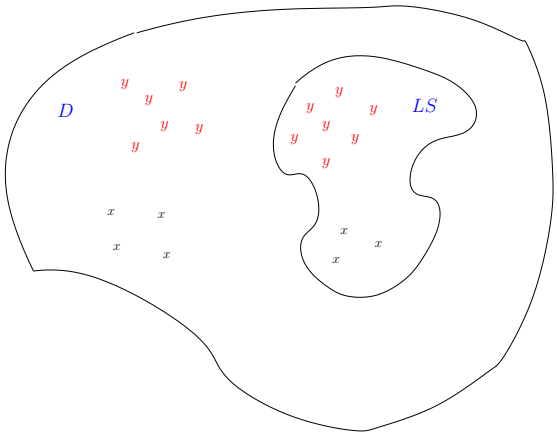

Fig. 4.2 – *y est un vrai positif.*

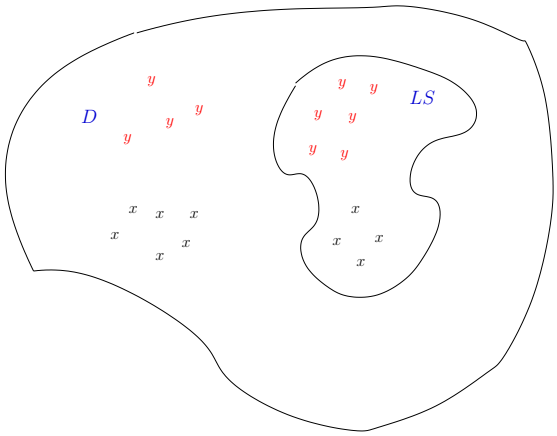

Fig. 4.3 – *y est un faux positif.*

– y est un vrai négatif (noté VN), c'est-à-dire qu'il n'est fréquent ni dans LS ni selon D. Par exemple, dans la figure 4.5, y a un support de 50% dans LS, et de 45% dans D.

Ces quatre situations sont résumées dans le tableau 4.1, où $\bar{S}$ (resp. $\bar{D}$) représentent les éléments non fréquents dans S (resp. D).

On voit alors bien par ce tableau que le fait d'assurer la correction et la complétude sur l'ensemble LS (c'est-à-dire trouver tous les motifs fréquents de LS et uniquement ceux-ci) n'assure pas la correction et la complétude par rapport à la distribution sous-jacente D (certes, inconnue). Comme nous l'avons illustré, certains éléments peuvent être à tort ajoutés (FP) ou oubliés (FN). Avant d'aborder ce problème d'un point de vue formel, nous allons d'abord montrer par des expérimentations que la taille de l'échantillon LS a, et ce de manière logique, une forte influence sur la qualité des résultats.

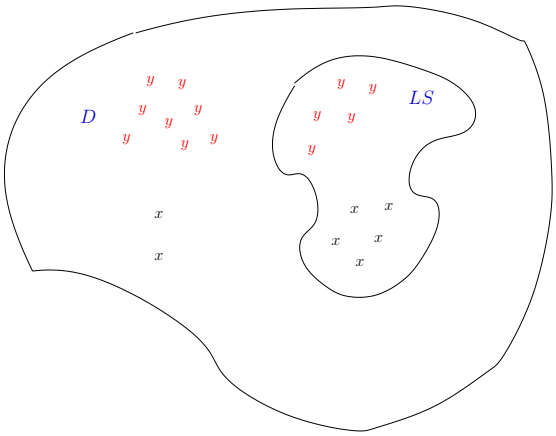

FIG. 4.4 – *y est un faux négatif.*

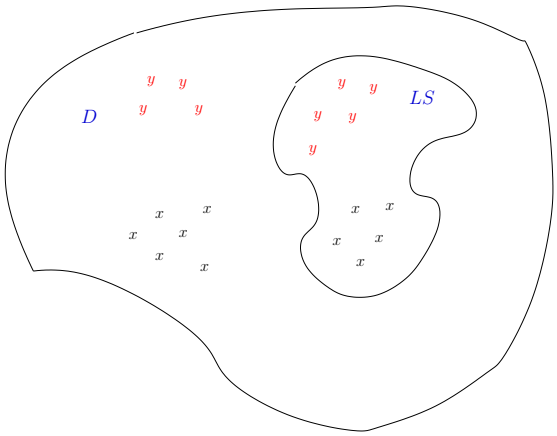

FIG. 4.5 – *y est un vrai négatif.*

	S	$\bar{S}$
D	VP	FN
$\bar{D}$	FP	VN

TAB. 4.1 – *Les situations possibles en fouille de données séquentielles.*

4.2.3 Étude expérimentale préliminaire

Pour effectuer cette expérimentation, nous avons choisi de travailler sur la base ATIS (Air Travel Information System) qui est composée de 14044 phrases en langue anglaise. Chaque mot de la phrase constitue un item (un élément de l'alphabet). Ce sont des phrases "parlées" contenant des demandes d'informations à propos du trafic aérien, des billets d'avions, des horaires, etc.

Pour simplifier, nous considérons la base contenant les 14044 séquences comme étant la distribution sous-jacente D. Nous considérons donc les sous-séquences issues de cet ensemble comme l'ensemble correct et complet à atteindre. Le but est donc d'extraire les motifs fréquents dans une sous-base ATIS de taille croissante et de les comparer avec les motifs extraits sur la base complète.

Protocole Expérimental

Nous avons choisi arbitrairement un support de 10%, à partir duquel nous avons extrait les motifs fréquents de la base ATIS complète (nous avons utilisé l'algorithme SPAM présenté au chapitre 1.3.2). Notons MF cet ensemble qui servira de référence pour le calcul des erreurs. Nous avons ensuite construit des échantillons LS_i de taille croissante (de 10 à 14044). Pour chaque échantillon LS_i, SPAM extrait l'ensemble des motifs fréquents, noté MF_i. Nous avons calculé ensuite, pour chaque échantillon LS_i, le nombre de faux positifs, c'est-à-dire le nombre de motifs présents dans MF_i et non présents dans MF, et le nombre de faux négatifs, c'est-à-dire présents dans MF et pas

dans MF_i. À partir de ces quantités, nous pouvons calculer les taux de rappel et de précision. En effet, si nous reprenons le tableau 4.1, le rappel correspond au nombre de motifs réellement fréquents retrouvés dans MF_i, par rapport au nombre total de motifs fréquents dans MF, soit :

$$rappel = \frac{VP}{VP + FN}.$$

La précision correspond au nombre de motifs réellement fréquents retrouvés dans MF_i, par rapport au nombre total de motifs trouvés dans MF_i, soit :

$$precision = \frac{VP}{VP + FP}.$$

Notons que, pour chaque taille d'échantillon, 15 expériences sont effectuées et tous les résultats sont moyennés.

Résultats

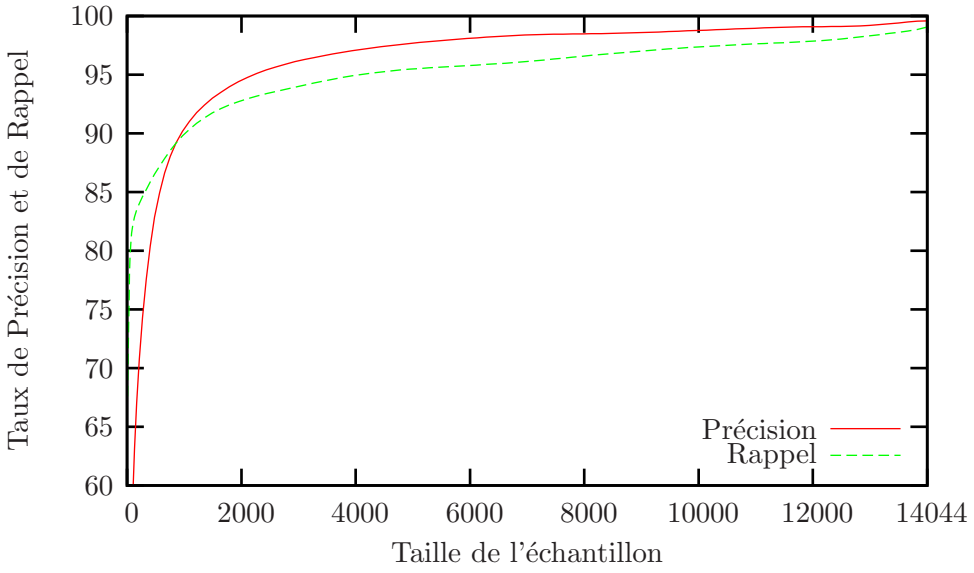

Fig. 4.6 – *Évolution des taux de rappel et de précision selon la taille de l'échantillon.*

Les courbes de précision et rappel de la figure 4.6 montrent clairement que plus la taille de l'échantillon est grande et plus les résultats obtenus, par l'algorithme de fouille de données séquentielles, sont proches des résultats obtenus sur la totalité des données. Par exemple, pour un échantillon de taille 2000, environ 94% des motifs fréquents trouvés sont des vrais fréquents (précision) et 92% des motifs réellement fréquents sur la base entière sont retrouvés (rappel). De plus, nous remarquons que les courbes ont un comportement asymptotique ce qui nous permet de penser qu'il est possible de chercher une taille d'échantillon nécessaire et suffisante permettant de trouver "sans trop

de perte" des résultats identiques à ceux trouvés si nous disposions de la distribution cible, de taille potentiellement infinie.

Pour conforter nos positions, nous avons également calculé les différences de support des motifs extraits sur la base ATIS et sur les échantillons LS_i. Ce calcul est fait sur l'ensemble des vrais positifs, c'est-à-dire les motifs communs à MF et MF_i. Les résultats sont normalisés par le nombre de vrais positifs et moyennés sur les mêmes 15 expériences que précédemment. La courbe de la figure 4.7 montre le même comportement asymptotique que les précédentes. Plus la taille de LS_i est proche de la taille de la base ATIS, plus les taux de supports calculés sont proches de la réalité.

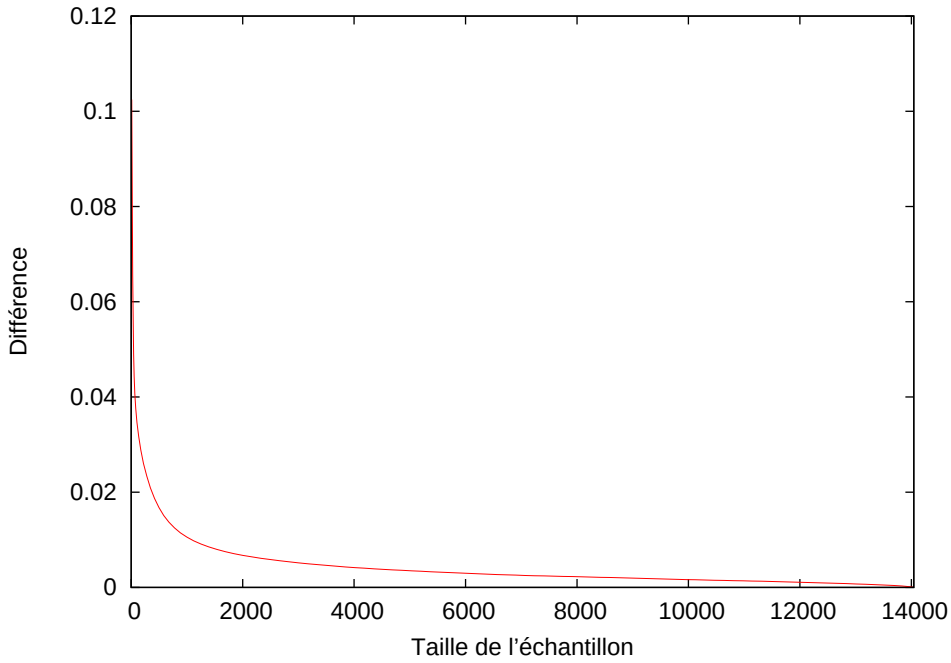

FIG. 4.7 – *Évolution des différences de supports en fonction de la taille de l'échantillon.*

Les constats faits ici expérimentalement nous incitent à étudier plus formellement les influences de la taille de l'échantillon LS sur la qualité des résultats obtenus. L'objectif est de mettre en place une borne sur la taille des données nécessaire pour assurer des taux de précision et de rappel donnés.

4.2.4 Borne sur le nombre de séquences

Nous avons montré précédemment que la fouille de données séquentielles *classique*, c'est-à-dire dont les critères de correction et de complétude sont basés sur le seul ensemble de séquences LS ne constitue pas une solution satisfaisante. La distribution cible D dont est issu l'échantillon doit être prise en compte dans le processus de fouille ou dans l'évaluation des résultats. Or, comme nous l'avons précisé cette distribution

est inconnue. Nous proposons donc de projeter le problème de la recherche de motifs séquentiels fréquents dans un cadre de tests d'hypothèses statistiques.

Test de proportion

Notons $\hat{p}(w)$ la proportion de séquences dans l'échantillon LS qui contiennent le motif w, où $w =< x_1, \ldots, x_l >$ avec $x_1, \ldots x_l$ des items pas forcément consécutifs dans les séquences. Le cœur des algorithmes de recherche de motifs fréquents est basé sur le test du support qui consiste à comparer la fréquence d'un motif à un support minimal fixé par l'utilisateur. D'un point de vue statistique inférentielle, nous pouvons traduire cette opération par un test de proportion, caractérisé par une hypothèse nulle H_0 et une hypothèse alternative H_a. Puisque LS est issue de la distribution cible D, $\hat{p}(w)$ ne constitue qu'une estimation de la probabilité réelle $p(w)$. Afin de vérifier si le motif w est réellement fréquent (*i.e.* supérieur au seuil de support fixé, noté p_0), nous devons poser l'hypothèse nulle suivante :

$$H_0 \ : \ p(w) \geq p_0.$$

L'hypothèse alternative décrit le fait que le motif w n'est en réalité pas fréquent selon la distribution D, soit

$$H_a \ : \ p(w) < p_0.$$

Lorsqu'un test statistique est effectué, deux types d'erreurs peuvent apparaître. Ces erreurs sont dues au fait que, la distribution cible étant inconnue, $p(w)$ doit être estimé par $\hat{p}(w)$. La première erreur, appelée erreur de Type I et généralement notée α, correspond au fait de rejeter à tort H_0. La seconde erreur, appelée erreur de Type II et généralement notée β, correspond au fait d'accepter à tort l'hypothèse H_0. Analysons d'un peu plus près ces deux erreurs.

Erreur de Type I α représente le risque de rejeter H_0 alors que l'hypothèse est vraie. Par conséquent, dans notre contexte de la fouille de données séquentielles, α correspond au risque de rejeter un vrai motif fréquent, c'est-à-dire un motif fréquent selon la distribution D. Ce risque α est donc directement lié à la proportion de faux négatifs, vus précédemment. Il est possible de contrôler α en calculant une borne de rejet de H_0, notée k, au delà de laquelle il ne reste que $\alpha\%$ de chance que w soit réellement fréquent (*i.e.* validant en réalité H_0). Fixons donc l'erreur de Type I à une valeur α donnée. Nous obtenons donc :

$$\alpha = P\left(\hat{p}(w) < k | H_0 \ vraie\right). \tag{4.1}$$

Le nombre de séquences de LS qui contiennent la sous-séquence w est une variable aléatoire binomiale ayant une probabilité de succès de $p(w)$. Quand le nombre de séquences N présentes dans l'échantillon est supérieur à 30, il est connu en statistique inférentielle, grâce au théorème central limite, que la proportion $\hat{p}(w)$ suit une distribution normale $\mathcal{N}$. Nous supposerons dans la suite que cette contrainte $N > 30$ est intrinsèquement vérifiée, puisque notre but est de trouver une borne théorique sur N.

Les expérimentations que nous présenterons dans la suite confirmeront que cette hypothèse n'est pas forte. Dans ce contexte, $\hat{p}(w)$ suit une loi normale centrée sur $p(w)$ et d'écart type $\sqrt{\frac{p(w)(1-p(w))}{N}}$ soit :

$$\hat{p}(w) \approx \mathcal{N}\left(p(w), \sqrt{\frac{p(w)(1-p(w))}{N}}\right).$$

L'équation 4.1 peut donc être réécrite de la façon suivante :

$$\alpha = P\left(\frac{\hat{p}(w) - p(w)}{\sqrt{\frac{p(w)(1-p(w))}{N}}} < \frac{k - p(w)}{\sqrt{\frac{p(w)(1-p(w))}{N}}} | H_0 \ vraie\right). \tag{4.2}$$

Comme H_0 est vraie, nous pouvons remplacer $p(w)$ par une valeur satisfaisant H_0. Puisque nous sommes intéressés par le rejet de H_0, nous pouvons fixer $p(w)$ à la valeur minimale conservant l'hypothèse vraie, soit en fait le support p_0. Nous obtenons ainsi :

$$\alpha = P\left(\frac{\hat{p}(w) - p_0}{\sqrt{\frac{p_0(1-p_0)}{N}}} < \frac{k - p_0}{\sqrt{\frac{p_0(1-p_0)}{N}}}\right). \tag{4.3}$$

Nous pouvons alors facilement déduire la borne k satisfaisant un risque α :

$$k = p_0 - z_\alpha \sqrt{\frac{p_0(1-p_0)}{N}}, \tag{4.4}$$

où z_α correspond au $(1-\alpha)$ percentile de la distribution normale. Pour récapituler, l'équation 4.4 nous donne la borne de rejet de H_0 garantissant une erreur de Type I égale à α. La règle de décision est donc la suivante : *si* $\hat{p}(w) < k$ *alors le motif* w *est considéré comme non fréquent*. Par exemple, supposons que nous disposons d'un échantillon de $N = 10000$ séquences et que nous fixons le seuil de support à $p_0 = 10\%$ et un risque $\alpha = 5\%$. En lisant la table de la distribution normale, on obtient $z_\alpha = 1.645$. En utilisant ces valeurs dans l'équation 4.4 nous obtenons

$$k = 0.1 - 1.645 \times \sqrt{\frac{0.1 * 0.9}{10000}} = 0.095.$$

Par conséquent, pour contrôler le risque α de rejeter à tort un motif réellement fréquent dans D, on acceptera tous les motifs dans LS apparaissant dans plus de 9.5% des séquences. Et ce, même si le support a été fixé à 10% !

Erreur de Type II Concernant β, il décrit le risque d'accepter H_0, alors que c'est l'hypothèse alternative H_a qui est vraie. Dans le contexte de la fouille de données séquentielles, cela signifie que β décrit le risque d'accepter un faux positif, c'est-à-dire un faux fréquent. Contrairement à α, la valeur de β peut être calculée en fonction de la borne k. Plus précisement, posons l'erreur de Type II :

$$\beta = P(\hat{p}(w) > k | H_a \ vraie). \tag{4.5}$$

Comme nous l'avons fait précédemment pour α, nous pouvons appliquer le théorème central limite et obtenons :

$$\beta = P\left(\frac{\hat{p}(w) - p(w)}{\sqrt{\frac{p(w)(1-p(w))}{N}}} > \frac{k - p(w)}{\sqrt{\frac{p(w)(1-p(w))}{N}}} | H_a \ vraie\right). \tag{4.6}$$

Pour calculer β, il est nécessaire de poser une valeur pour $p(w)$ vérifiant H_a. Comme l'hypothèse $H_a : p(w) < p_0$ est vraie, nous devons fixer une valeur inférieure à p_0. Soit p_a cette valeur (voir la section 4.2.5 pour une discussion sur la valeur de p_a). En remplaçant $p(w)$ par sa valeur sous H_a, nous obtenons :

$$\beta = P\left(\frac{\hat{p}(w) - p_a}{\sqrt{\frac{p_a(1-p_a)}{N}}} > \frac{k - p_a}{\sqrt{\frac{p_a(1-p_a)}{N}}}\right). \tag{4.7}$$

Comme k est connu grâce à l'équation 4.4, la partie droite de l'inégalité peut être calculée, et il suffit de lire dans la table de la distribution normale le β correspondant au z_β obtenu. Si nous reprenons l'exemple précédent (avec $N = 10000$ séquences), supposons que $p_a = 9\%$:

$$\frac{k - p_a}{\sqrt{\frac{p_a(1-p_a)}{N}}} = \frac{0.095 - 0.09}{\sqrt{\frac{0.09(1-0.09)}{10000}}} = 1.747.$$

En lisant dans la table de la loi normale, on obtient $\beta = 4\%$. Par conséquent, alors qu'il y avait un risque $\alpha = 5\%$ d'avoir un faux négatif, il y a 4% de risque d'accepter un faux positif, c'est-à-dire dans notre exemple, qui aurait une fréquence supérieure à 10% dans LS mais une probabilité réelle sous D de seulement 9%.

Borne minimale sur N L'objectif d'un processus de fouille de données séquentielles pertinent devrait donc toujours être de réduire à la fois les risques α et β. Malheureusement, il existe un système de vases communicants entre ces deux erreurs. Avec un nombre fixé de séquences N, la diminution de α fait augmenter β et vice-versa. Ce phénomène est clairement illustré dans la figure 4.8. Cette courbe représente les lois suivies par $\hat{p}(w)$ sous l'hypothèse H_0, d'espérance p_0, et sous l'hypothèse H_a, d'espérance p_a. α correspond à la densité de probabilité de la distribution de Laplace-Gauss correspondant à H_0 en dessous de la borne de rejet k. β correspond à la densité de probabilité de la distribution de Laplace-Gauss correspondant à H_a au dessus de la borne de rejet k. Sous cette configuration, il est clair que, pour faire diminuer la valeur de α il faut faire diminuer la borne de rejet k, ce qui implique inévitablement l'augmentation de la valeur de β. À taille d'échantillon fixée N, il est donc impossible de faire réduire à la fois les valeurs de α et β.

Notre solution est donc, de ne plus considérer cette taille d'échantillon comme étant fixée mais plutôt de chercher la taille minimale de LS permettant de ne pas dépasser des taux d'erreurs fixés α et β. L'augmentation de la taille N aura pour effet la diminution

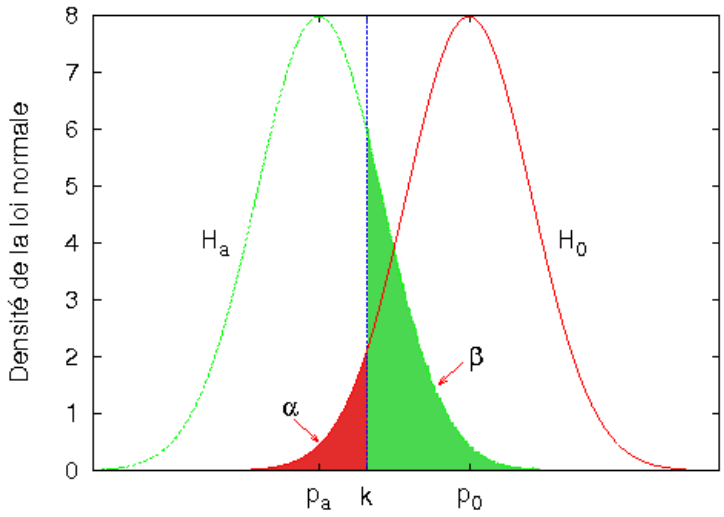

FIG. 4.8 – *Compromis entre les erreurs de Type I et II.* p_0 *(resp.* p_a*) est l'espérance de* $\hat{p}(w)$ *sous* H_0 *(resp.* H_a*).*

de l'écart type de la distribution normale et donc la diminution des valeurs des deux erreurs. Ceci est illustré à la figure 4.9, qui comparée à la figure 4.8 montre clairement que les quantités α et β sont plus petites.

Théorème 4.1 *Pour assurer que les proportions de faux négatifs et de faux positifs n'excèdent pas respectivement des risques fixés* α *et* β*, le nombre minimal de séquences* N_{low} *sur lequel l'algorithme de fouille de données séquentielles doit être exécuté est égal à :*

$$N_{low} = \left[\frac{z_\beta \sqrt{p_a(1-p_a)} + z_\alpha \sqrt{p_0(1-p_0)}}{p_0 - p_a} \right]^2, \ \forall p_0, p_a \in [0,1], p_a < p_0.$$

Preuve : Nous pouvons déduire de l'équation 4.7 que :

$$k = p_a + z_\beta \sqrt{\frac{p_a(1-p_a)}{N}}, \tag{4.8}$$

où z_β est le $(1-\beta)$ percentile de la distribution normale. Par identification des termes des équations 4.4 et 4.8, nous pouvons déduire que :

$$p_0 - z_\alpha \sqrt{\frac{p_0(1-p_0)}{N}} = p_a + z_\beta \sqrt{\frac{p_a(1-p_a)}{N}}. \tag{4.9}$$

En extrayant N de cette équation (4.9) nous obtenons la borne minimale présentée dans le théorème 4.1. □

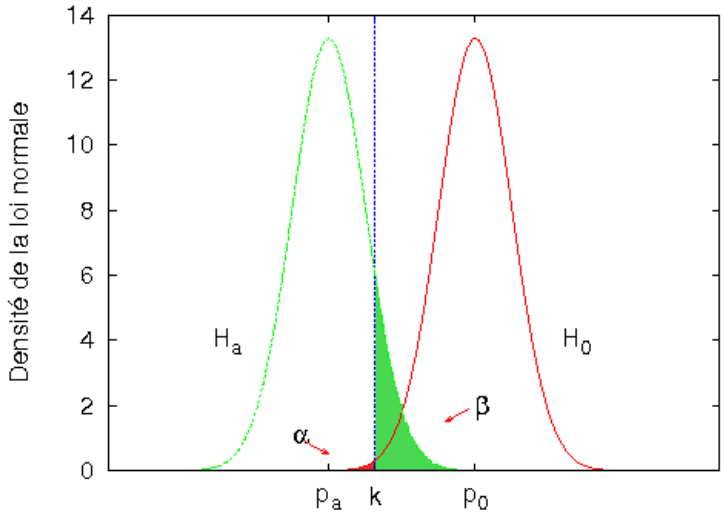

Fig. 4.9 – *Effet de la taille N sur les valeurs des erreurs α et β.*

En reconsidérant la figure 4.8, nous pouvons analyser l'impact de la valeur de p_a sur l'erreur β. Plus p_a s'éloigne de p_0 et plus β diminue. De même que la réduction de α et β implique l'augmentation de N, la réduction de β et p_a implique l'augmentation de N.

Pour illustrer cette borne minimale, les courbes de la figure 4.10 montrent l'évolution de N_{low} en fonction des taux d'erreurs α et β, du seuil de support p_0 et de p_a. Pour des questions de lisibilité nous avons choisi $\alpha = \beta$. Nous montrons deux courbes avec des valeurs différentes pour p_a. Pour assurer des taux d'erreurs fixés identiques de α et β (avec un seuil de support fixé), plus l'écart entre p_0 et p_a sera petit et plus le nombre de séquences nécessaires sera grand. Nous pouvons aussi remarquer que le seuil de support a une influence sur N_{low}. Pour un même écart $p_0 - p_a$, plus p_0 sera proche de 50% et plus le nombre de séquences nécessaires pour assurer des taux d'erreurs fixés sera grand (nous approfondirons ce constat plus tard). De plus, plus les seuils acceptables d'erreurs (α et β) sont petits et plus le nombre de séquences nécessaires augmente.

Illustration

Reprenons l'expérimentation sur la base ATIS présentée à la section 4.2.3. Pour illustrer notre borne, reprenons pour chaque échantillon LS_i, les risques empiriques $\hat{\alpha}$ et $\hat{\beta}$, correspondant respectivement à `1-rappel` et à `1-précision`. Il est intéressant de les comparer aux risques théoriques pour vérifier la pertinence de notre borne. Pour effectuer ceci, nous calculons la borne N_{low} pour des paramètres théoriques donnés α, β et p_a (p_0 étant fixé par l'application et égal à 10%). Par exemple, considérons

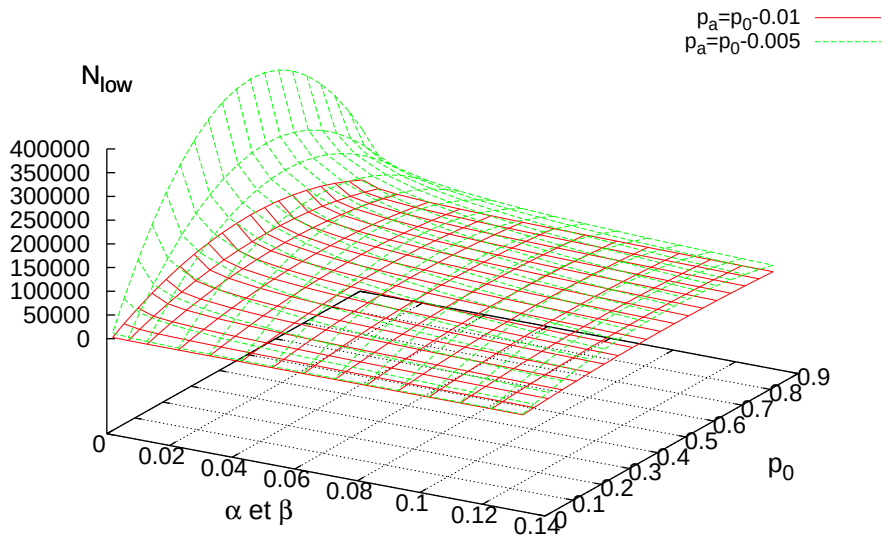

FIG. 4.10 – N_{low} *en fonction de* α, β, p_0 *et* p_a.

$\alpha = \beta = 5\%$ et $p_a = 9\%$. Instancions ces valeurs dans notre borne :

$$N_{low} = \left[\frac{1.645\sqrt{0.09(1-0.09)} + 1.645\sqrt{0.1(1-0.1)}}{0.1 - 0.09} \right]^2 = 9242.$$

Si nous observons, sur la figure 4.11, le résultat obtenu pour 9242 séquences, nous pouvons conclure que notre borne est pertinente puisque les deux erreurs calculées sur la base ATIS ($\hat{\alpha} \simeq 2\%$ *et* $\hat{\beta} \simeq 3\%$) n'excèdent pas les valeurs théoriques a priori fixées ($\alpha = 5\%$ *et* $\beta = 5\%$) pour le calcul de N.

4.2.5 Discussion sur la valeur de p_a

Jusqu'à présent, pour illustrer notre borne N_{low}, nous avons utilisé une valeur pour p_a assez proche de p_0. Par exemple, pour la base ATIS, nous avons choisi $p_a = p_0 - 1\%$. Rappelons que p_a est l'espérance de $\hat{p}(w)$ sous l'hypothèse alternative H_a et a pour seule contrainte d'être inférieure à p_0. Comme nous l'avons déjà précisé, il y a une dépendance forte entre p_a et notre borne basse N_{low}. Plus précisément, N_{low} croît quadratiquement avec l'augmentation de la valeur de p_a, c'est-à-dire avec la diminution de l'écart entre p_0 et p_a. Par conséquent, et même si la borne N_{low} n'est pas remise en cause, le choix d'une valeur pertinente pour p_a reste un important problème à aborder. Nous allons étudier dans ce qui suit trois solutions pour choisir la valeur de p_a.

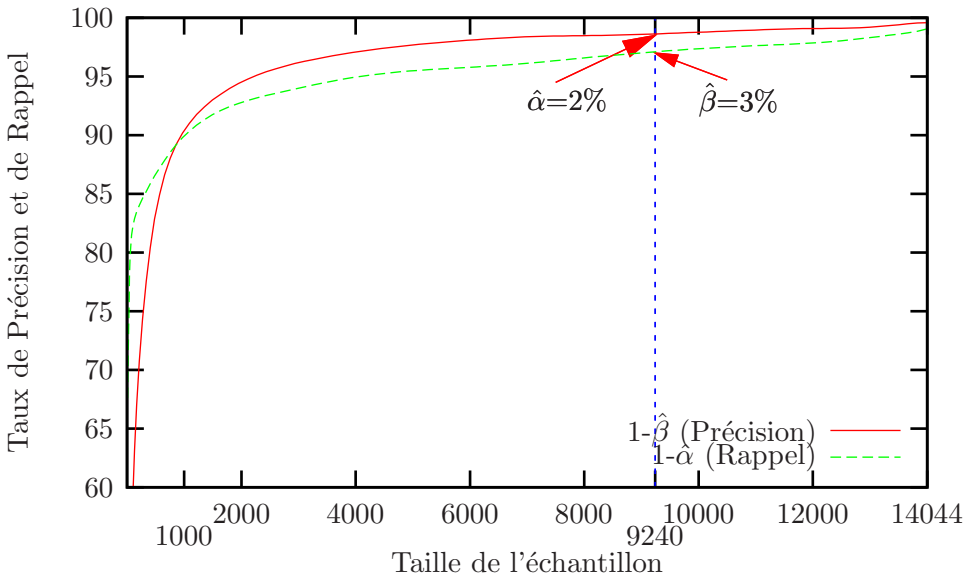

Fig. 4.11 – *Application de la borne N_{low} à un cas réel.*

Premièrement nous proposons une résolution d'un point de vue théorique. Ensuite une solution pratique est envisagée. Enfin, nous donnons une solution en moyenne.

Une solution théorique

La première solution, pour résoudre le problème de choix de valeur pour p_a est de considérer que w n'est plus un motif fréquent, très précisement à partir du moment où sa vraie probabilité sous la distribution D est plus petite que le seuil de support p_0. Soit N_0 le nombre de séquences tel que $p_0 = \frac{N_0}{N_{low}}$. Par conséquent, une sous-séquence w ne sera plus fréquente dès qu'elle apparaîtra moins de $N_0 - 1$ fois dans les N_{low} séquences. Nous obtenons donc :

$$p_a = \frac{N_0 - 1}{N_{low}} = p_0 - \frac{1}{N_{low}}.$$

En remplaçant p_a par cette valeur dans l'équation 4.8 nous obtenons :

$$k = p_0 - \frac{1}{N_{low}} + z_\beta \sqrt{\frac{(p_0 - \frac{1}{N_{low}})(1 - p_0 - \frac{1}{N_{low}})}{N_{low}}} \tag{4.10}$$

En exploitant cette équation et l'équation 4.4 nous obtenons une nouvelle formule analytique pour la borne basse, sous la forme d'un polynôme de degré 4 dont les solutions

donnent N_{low} (ce polynôme a été obtenu en utilisant MAPLE^{TM}).

$$\begin{aligned}
& (-2z_\beta^2 z_\alpha^2 p_0^2 + z_\beta^4 p_0^4 + 4z_\beta^2 p_0^3 z_\alpha^2 + z_\beta^4 p_0^2 - 2z_\beta^2 p_0^4 z_\alpha^2 + z_\alpha^4 p_0^4 + z_\alpha^4 p_0^2 - 2z_\alpha^4 p_0^3 - 2z_\beta^4 p_0^3)\mathbf{N^4} \\
+ \quad & (2p_0(-z_\alpha^2 - z_\beta^4 + z_\beta^2 p_0 - z_\beta^2 + z_\alpha^2 z_\beta^2 + z_\alpha^2 p_0) + 4p_0^3(z_\beta^2 z_\alpha^2 - z_\beta^4) + 6p_0^2(z_\beta^4 - z_\beta^2 z_\alpha^2))\mathbf{N^3} \\
+ \quad & (1 - 4z_\beta^2 p_0 + 2p_0 z_\alpha^2 z_\beta^2 + 2z_\beta^2 - 2z_\beta^2 z_\alpha^2 p_0^2 + z_\beta^4 - 6z_\beta^4 p_0 + 6z_\beta^4 p_0^2)\mathbf{N^2} \\
+ \quad & (2z_\beta^4 + 2z_\beta^2 - 4z_\beta^4 p_0)\mathbf{N} + z_\beta^4 = 0.
\end{aligned} \tag{4.11}$$

Comme idéalement souhaité, nous constatons à présent que la borne ne dépend plus que des risques α et β, et du seuil de support p_0. N_{low} ne dépend plus de p_a, et les solutions de l'équation donnent une borne minimale exacte garantissant des taux d'erreurs α et β pour un seuil de support donné. Cependant, elle constitue une réponse extrêmement pessimiste à notre problème. En effet, en résolvant cette équation pour des paramètres qui pourraient classiquement être utilisés dans un problème de fouille de données séquentielles ($\alpha = \beta = 5\%$ et $p_0 = 10\%$), nous obtenons $N_{low} = 3.10^{20}$ séquences ! Il est clair que cette solution ne sera pas utilisable en pratique et constitue une borne bien trop pessimiste. Ceci est logique au vu de la valeur choisie pour p_a, qui est elle même la plus pessimiste possible. En effet, en travaillant de la sorte, nous avons considéré que tous les faux positifs acceptés sur LS (alors qu'ils auraient dû être refusés) n'étaient en réalité présents que dans $p_0 - \frac{1}{N_{low}}$ pourcents des cas selon D, ce qui est hautement improbable en pratique.

Une solution pratique

Une solution plus réaliste, et plus pratique, est celle utilisée dans nos précédentes expérimentations. Comme p_0 est souvent exprimé en pourcentage d'occurrences de w dans l'ensemble de séquences, une solution alternative peut consister à choisir une valeur pour p_a égale à $p_0 - 1\%$. Pour confirmer la pertinence de cette stratégie, nous avons calculé la borne théorique N_{low} (avec la formule du théorème 4.1) pour différentes valeurs de risques α et β. Ces résultats théoriques ont ensuite été comparés à différentes courbes de rappels et précisions empiriques calculées à partir de 10 bases différentes. Les quatre premières sont des bases de données réelles :

- ATIS que nous avons déjà utilisée dans les expériences précédentes (14044 phrases en langue anglaise, où les items sont les mots des phrases),
- FIRSTNAMES est un ensemble de 12437 prénoms masculins et féminins de pays d'origines différents,
- FRENCH WORDS est une base de 250750 mots de la langue française (sans doublons),
- TOWNS est une base de 39074 noms de villes françaises.

FIRSTNAMES, FRENCH WORDS et TOWNS sont issues de dictionnaires de l'ABU que l'on peut trouver à l'adresse `http://abu.cnam.fr/`. Nous avons aussi construit 6

bases artificielles à partir d'automates probabilistes :

- Reber est une base de 15000 séquences issues de la grammaire régulière de Reber [Reb67] dont la distribution cible est un automate constitué de 8 états et d'un alphabet de 7 lettres (voir l'automate de la figure 4.12).
- $SxLy$ sont aussi des bases de 15000 séquences générées à partir d'automates à x états et y lettres. Nous avons généré 5 bases en faisant varier le nombre d'états et de lettres.

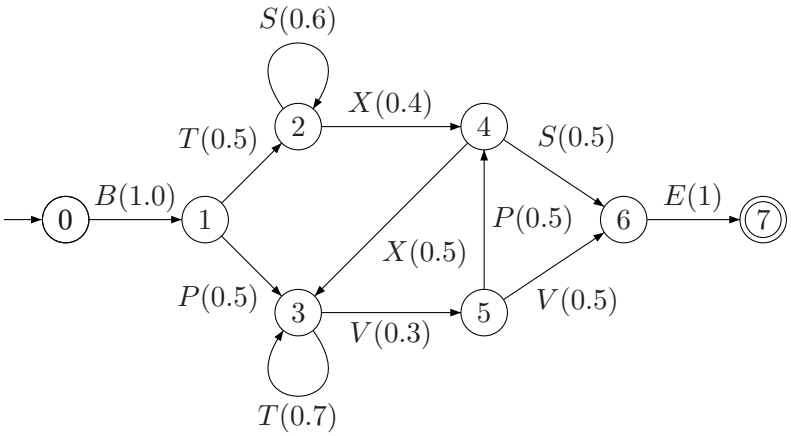

Fig. 4.12 – pdfa *correspondant à la grammaire de* Reber.

Notons que, dans le cas des bases de données artificielles, la distribution théorique D est connue sous la forme d'un automate à partir duquel ont été générées les données. Il est donc possible de calculer la probabilité de n'importe quel motif w, en utilisant des méthodes mathématiques appropriées comme celles présentées au chapitre 3.

Pour chacune de ces bases, nous avons utilisé le même protocole expérimental que celui présenté à la section 4.2.3. Avec l'algorithme spam, nous avons calculé pour chacune des bases prises dans leur totalité, l'ensemble des motifs fréquents. Le seuil de support utilisé est de $p_0 = 10\%$. Cet ensemble constitue la référence théorique. Ensuite, nous avons construit des échantillons de tailles croissantes (de 10 à 15000) à partir desquels nous avons aussi extrait les motifs fréquents. Nous calculons ensuite les valeurs empiriques de rappel $(1 - \hat{\alpha})$ et de précision $(1 - \hat{\beta})$.

Les résultats sont respectivement présentés sur les figures 4.13 et 4.14 avec les courbes théoriques obtenues (en rouge). Plusieurs remarques peuvent être faites. Premièrement, ces courbes confirment que notre approche pragmatique consistant à prendre $p_a = p_0 - 1\%$ est satisfaisante puisque nous fournissons bien une borne sur le nombre de séquences nécessaires pour garantir au minimum des taux de rappels et précisions théoriques a priori fixés. Quelle que soit la base de données, les courbes correspondantes sont, en effet, toujours au dessus des valeurs théoriques. Deuxièmement, notons que la distance entre les risques empiriques et notre borne minimale est assez grande pour la plupart des bases, et en particulier pour toutes les bases réelles. Cela signifie que notre borne reste quand même assez pessimiste, et que dans le cas des bases réelles,

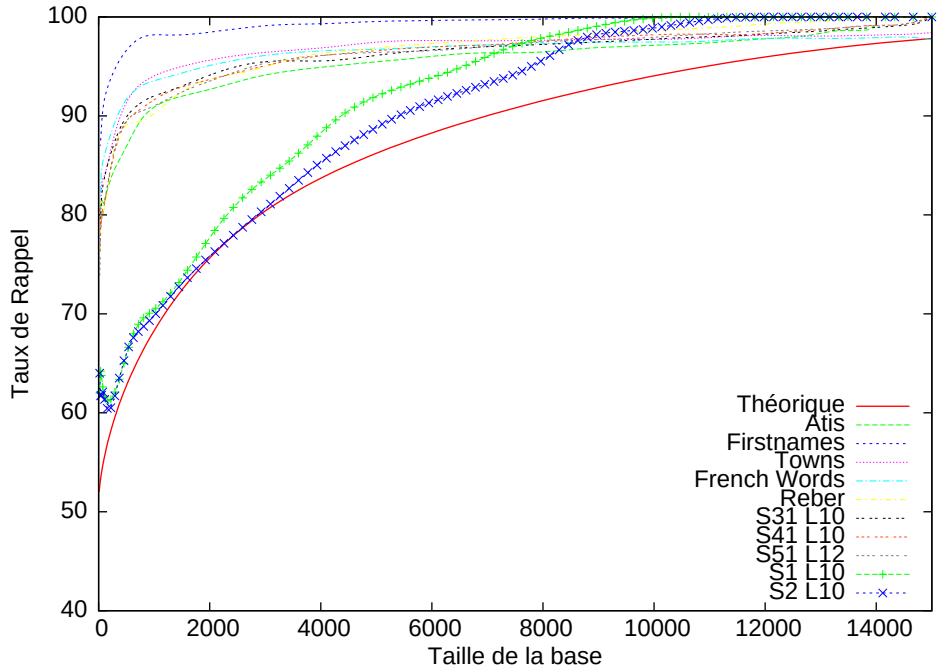

FIG. 4.13 – *Comparaison entre différents rappels empiriques et* $1-\alpha$*, pour* $p_0 = 0.1$.

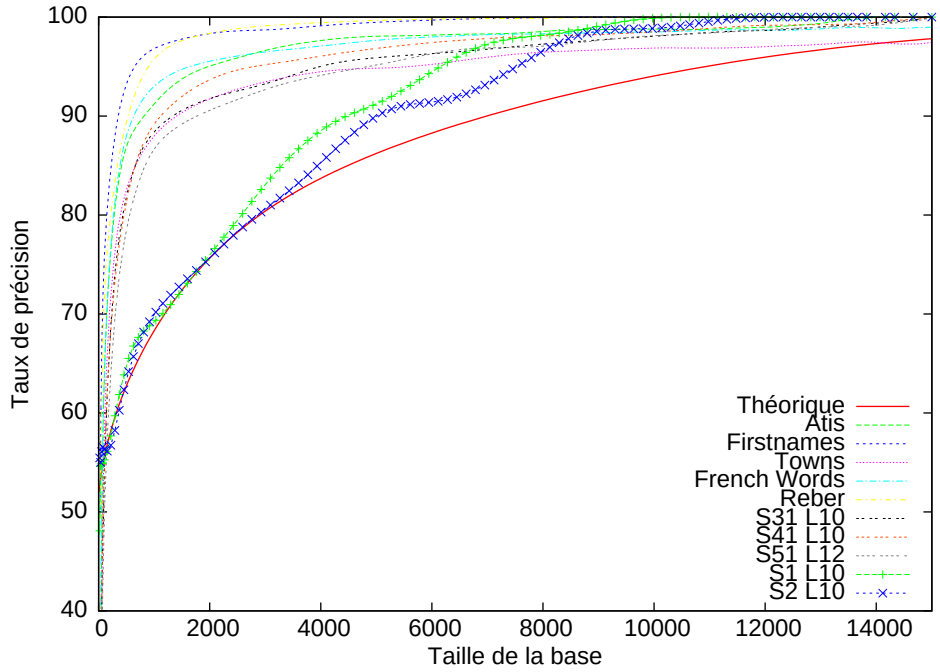

FIG. 4.14 – *Comparaison entre différentes précisions empiriques et* $1-\beta$*, pour* $p_0 = 0.1$.

une valeur de p_a moins permissive, c'est-à-dire plus proche de p_0 aurait donné des résultats encore meilleurs. Cependant, ce constat ne remet pas en cause la pertinence de notre borne puisque, comme le montrent les courbes artificielles obtenues à partir des automates $S1L10$ et $S2L10$, il peut arriver que les risques empiriques soient beaucoup plus proches des valeurs théoriques. L'effet d'échantillonnage ou encore la nature des distributions cibles peuvent donc avoir des effets importants.

Notons que les courbes théoriques décrites aux figures 4.13 et 4.14 ne traitent que le cas de $p_0 = 10\%$. Dans le but de fournir un outil calculatoire rendant l'estimation du nombre de séquences minimum plus facile, nous avons construit des courbes théoriques pour différentes valeurs de p_0. Au vu des résultats précédents, nous avons choisi de conserver $p_a = p_0 - 1\%$ et pour faciliter les calculs nous avons posé $\alpha = \beta$. La figure 4.15

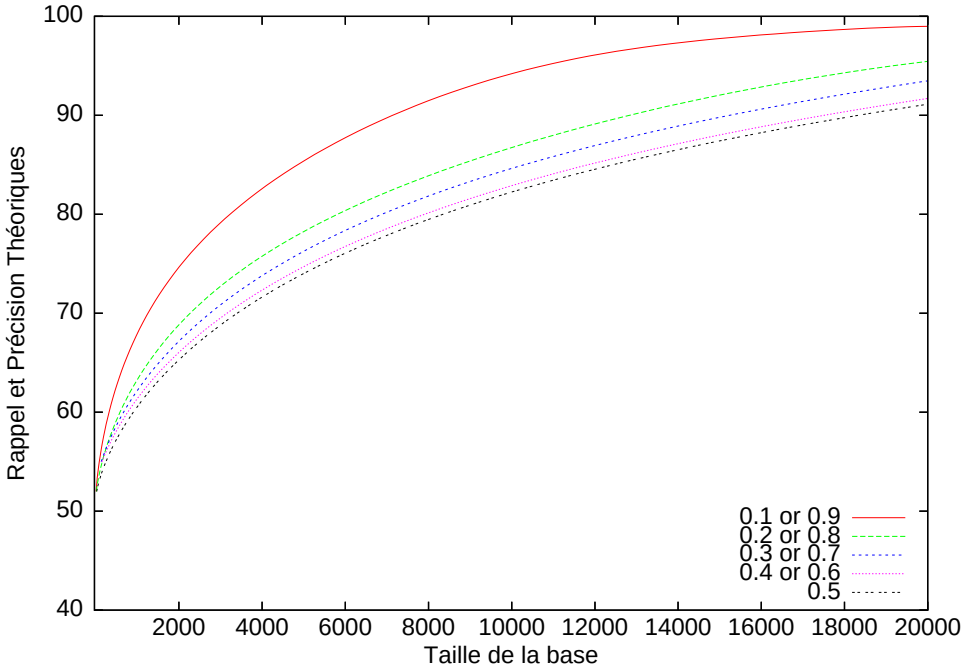

Fig. 4.15 – *Évolution des rappels et précisions théoriques en fonction de la borne minimale N_{low} et du seuil de support p_0 (de 0.1 à 0.9).*

décrit un ensemble d'abaques qui donnent directement le nombre minimal de séquences N_{low} nécessaires pour garantir au moins un rappel de $1 - \alpha$ et une précision de $1 - \beta$. Nous pouvons noter que les courbes ne sont pas identiques. Cela signifie que la valeur de p_0 a un impact direct sur la borne minimale. D'un point de vue mathématique, cela peut facilement être expliqué par le fait que p_0 est utilisé dans la formule de N_{low} dans une fonction concave de la forme $p_0(1 - p_0)$ qui est maximale pour $p_0 = 0.5$. Par conséquent, pour des valeurs identiques de α et β, si l'on choisit $p_0 = 0.5$, le nombre de séquences nécessaires sera plus important que pour toute autre valeur. Par exemple, pour assurer des risques de 10%, avec un support de 10% (ou 90%), environ 8000

séquences seront nécessaires, alors que pour un support de 50% il en faudrait 18000. La conclusion très intéressante à ce constat est qu'il pourra être conseillé aux utilisateurs de choisir des supports très petits (ou très grands) pour minimiser les risques d'obtenir des faux positifs et des faux négatifs.

Une solution en moyenne

Dans les deux solutions précédentes, nous avons fixé la valeur de p_a pour calculer la borne minimale. En fixant p_a, **nous supposons que tous les motifs acceptés à tort suivent une loi de probabilité centrée sur cette valeur de** p_a, ou autrement dit d'espérance mathématique égale à p_a. Or, si on reconsidère l'hypothèse alternative $H_a : p(w) < p_0$, de laquelle est issue p_a, celle-ci décrit n'importe quelle valeur comprise entre 0 et p_0. Tous les motifs dont le support est inférieur à p_0 sont couverts par l'hypothèse H_a. Nous pouvons donc être tentés d'utiliser toutes les valeurs de p_a inférieures strictement à p_0 (notons p_0^- la plus grande de ces valeurs) pour calculer une valeur moyenne pour la borne N_{low}. Sous l'hypothèse que les probabilités des motifs soient uniformément distribuées sur l'intervalle $[0,p_0[$, le calcul de la moyenne de N_{low} se fait en intégrant N_{low}, qui dépend de p_a, entre 0 et p_0^-, soit :

$$Moy(N_{low}) = \frac{1}{p_0^- - 0} * \int_0^{p_0^-} \left[\frac{z_\beta \sqrt{p_a(1-p_a)} + z_\alpha \sqrt{p_0(1-p_0)}}{p_0 - p_a} \right]^2 dp_a.$$

Cette moyenne ne dépend donc plus de p_a mais uniquement de α, β et p_0. Nous avons voulu comparer cette moyenne aux courbes obtenues avec notre solution pratique $p_a = p_0 - 1\%$. De la même façon que précédemment, nous avons posé $\alpha = \beta$ pour faciliter les calculs. Nous avons calculé $Moy(N_{low})$, pour $p_0 = 0.2$, en faisant varier α et β. Nous montrons à la figure 4.16 les résultats obtenus sur les seules bases réelles.

Le constat est que la solution en moyenne est clairement moins pessimiste que la solution pratique où $p_a = p_0 - 1\%$ (courbe théorique en vert) même si nous n'obtenons "plus" qu'une borne minimale **en moyenne**.

Une perspective intéressante à ce travail consisterait à tenir compte de l'information présente dans les données. En effet, ici nous avons supposé que les données étaient uniformément distribuées et nous avons donc utilisé cette hypothèse dans notre intégration sur les valeurs de p_a. Or, il serait intéressant de prendre en compte la distribution empirique des données sous LS et d'utiliser, pour chaque valeur de p_a, la quantité de motifs ayant ce support. Des travaux sont en cours sur cette question, où la principale difficulté vient de l'estimation de la distribution des motifs sous LS et le passage d'une distribution discrète à un cas continu.

4.2.6 Travaux connexes

D'autres approches ont aussi tenté de considérer les données de LS comme un échantillon d'une distribution statistique théorique inconnue. Ces travaux s'intéressent donc aussi généralement aux deux critères essentiels que nous avons présentés : *les faux*

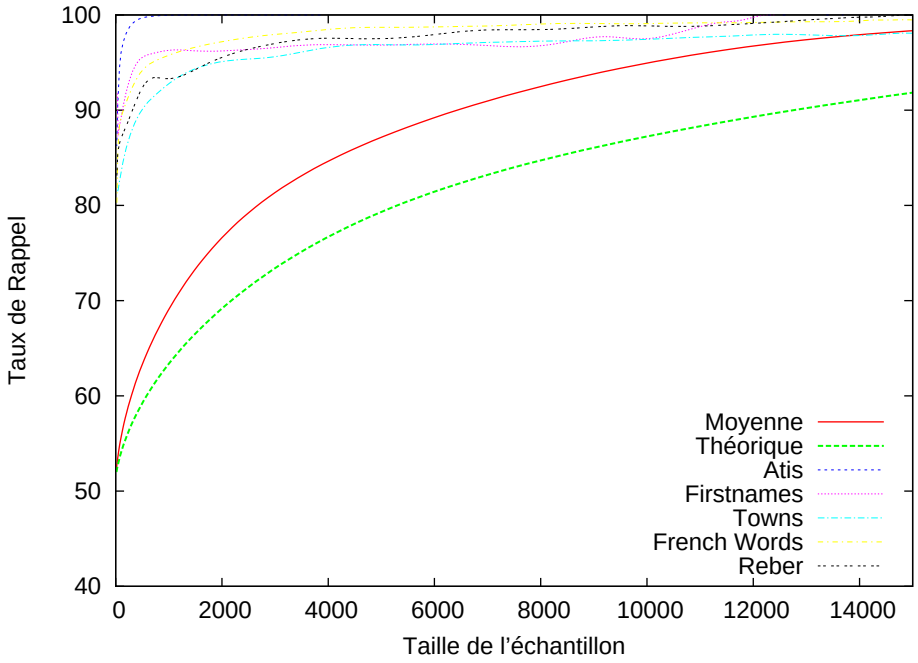

Fig. 4.16 – *Comparaison entre différents rappels empiriques et* $1-\alpha$*, pour* $p_0 = 0.2$.

positifs (ou encore β et 1− précision) et *les faux négatifs* (ou encore α et 1− rappel). Faisons un tour d'horizon de ces travaux.

Dans [Web07], l'auteur propose de contrôler le taux de faux positifs. Il propose deux méthodes qui utilisent des tests statistiques classiques, appliqués sur les motifs découverts, dans le but de contrôler le risque de découvrir des motifs qui ne sont pas réellement intéressants. A l'instar de ce qui est réalisé dans une tâche d'apprentissage, il propose tout d'abord de partager les données en deux ensembles, un ensemble *exploratoire* et un ensemble de *validation* (ou de test). Les motifs trouvés avec l'ensemble exploratoire sont ensuite évalués, avec l'ajustement de Bonferroni [Sha95], sur l'ensemble de test. Cet ajustement a été proposé pour contourner les problèmes de tests multiples. En effet, lorsqu'un test statistique est utilisé plusieurs fois au cours d'un processus, un problème se pose : par exemple, si α correspond au risque de prendre une unique mauvaise décision, répéter ce test plusieurs fois va augmenter ce risque [Sha95]. Pour contourner ce problème, plusieurs stratégies ont été proposées. Une des plus populaires est donc l'ajustement de Bonferroni qui utilise un risque $\frac{\alpha}{n}$ lorsque n tests d'hypothèses sont effectués. Cependant, lorsque n est grand, un tel ajustement est trop stricte et induit l'augmentation de l'autre risque β. Dans une deuxième approche, Webb propose une technique "d'ajustement direct", toujours par la méthode de Bonferroni, appliquée aux tests statistiques utilisés, au cours du processus de recherche des motifs, dans le but d'ajuster la taille de l'espace de recherche. Néanmoins, ces deux méthodes ne permettent pas de contrôler l'erreur de Type II.

Dans le cadre de la recherche de règles d'association, Meggiddo et Srikant [MS98] proposent une méthode, par une technique d'échantillonnage, permettant d'évaluer la quantité de règles d'association qui sont des fausses découvertes. Leur but est donc de contrôler la précision. Soit une règle $X \Rightarrow Y$, où X et Y sont des itemsets. Ils mettent en place un test statistique et considèrent comme hypothèse nulle $p(X \cup Y) = p(X) * p(Y)$ contre l'hypothèse alternative $p(X \cup Y) > p(X) * p(Y)$, ce qui signifie qu'une grande partie des transactions qui contiennent X contiennent aussi Y. Ils utilisent un test statistique pour exploiter la propriété que les fréquences observées d'un itemset suivent asymptotiquement une distribution normale. Pour réduire ce risque d'accepter une fausse découverte, ils augmentent le seuil de support p_0 par $z_\beta \times \sigma_{\hat{p}}$, où $\sigma_{\hat{p}}$ est l'écart type de $\hat{p}$ égal à $\sqrt{\frac{p_0(1-p_0)}{N}}$ et z_β est le $(1-\beta)$ percentile de la distribution normale. Par conséquent, en fixant a priori le risque β, ils peuvent contrôler la précision. Cependant, en utilisant des petites valeurs pour β, borner p_0 de cette façon produit une décroissance du rappel, et donc l'augmentation de l'erreur de Type I.

Dans [GMMT06], Gionis et al. proposent une méthode permettant d'assurer la pertinence des résultats obtenus par un processus de fouille dans des matrices booléennes. Leur idée est de vérifier que les résultats obtenus ne sont pas dûs au hasard. Pour cela, ils génèrent aléatoirement des matrices ayant les mêmes valeurs marginales (somme sur les lignes et les colonnes) que les données d'origine. Si les résultats sont identiques sur les données réelles et les données générées, cela signifie que les dépendances mises en évidence par l'algorithme ne sont pas réellement pertinentes.

Dans [ZPT04], Zhang et al. posent le problème de la découverte de règles SQ (statistiquement quantitatives) significatives dans des données comportant des attributs. Ils travaillent sur des règles modélisant des connaissances sur des parts de marché, c'est-à-dire du type : `les clients célibataires, habitants la région Nord Est, constituent` 25.15% `de part du marché`. Leur but est d'écarter les règles qui apparaissent uniquement par chance. Pour cela, ils mettent en place un intervalle de confiance sur la statistique considérée qui représente l'intervalle correspondant aux valeurs de la statistique apparaissant par chance. Pour calculer cet intervalle, ils utilisent un algorithme qui permute certaines valeurs des attributs. Dans cette approche, les règles qui ne sont pas découvertes alors qu'elles auraient dues, c'est-à-dire les faux négatifs, ne sont pas traitées.

Dans [LNSP07], Laur et al. s'attardent aussi sur le problème du contrôle des erreurs de Type I et II dans le cas des flux de données. Leur idée est d'annuler une des sources d'erreurs (choisie par l'utilisateur) avec une forte probabilité et de limiter l'autre. Pour ceci, ils proposent de modifier le seuil de support p_0 en utilisant les bornes de Chernoff. L'application de ces bornes dans le cadre d'une variable aléatoire binomiale (ce qui est le cas dans notre contexte) établit que l'erreur d'estimation d'une vraie probabilité peut être bornée comme suit :

$$\forall \epsilon \in]0,1[,\ P(|\hat{p} - p| \geq \epsilon) < e^{-2N\epsilon^2},$$

où p est la vraie probabilité d'un motif selon un distribution théorique inconnue. En posant cette quantité égale à une valeur fixée δ et en résolvant cette équation, pour

trouver ϵ, ils obtiennent une valeur de relâchement pour p_0. Les auteurs prouvent que pour obtenir une précision égale à 1 avec une probabilité de $1-\delta$, tout en contrôlant le rappel, il faut tester si $\hat{p} \geq p_0 + \epsilon$. Inversement, pour garantir un rappel égal à 1 avec une probabilité de $1-\delta$, tout en limitant la dégradation de la précision, Laur et al. testent si $\hat{p} \leq p_0 - \epsilon$.

Même si cette approche est théoriquement bien fondée, l'utilisateur doit choisir le critère (précision ou rappel) qu'il veut optimiser, ce qui peut être une tâche difficile dans les domaines où à la fois les erreurs de type I et II sont indésirables.

Comme nous pouvons le voir, il existe un certain nombre de travaux visant à étudier les erreurs commises par les algorithmes de fouille de données, mais, contrairement à notre approche, aucune ne permet réellement de contrôler à la fois les deux types d'erreurs (I et II), ce que nous avons pu réaliser en proposant notre borne sur N.

4.2.7 Conclusion

Nous avons proposé dans cette section une borne sur le nombre de séquences qu'il est nécessaire de fouiller pour contrôler le taux de faux positifs et de faux négatifs. La question cruciale qui demeure est la suivante : que faire lorsque le domaine d'application qui nous est proposé ne nous permet pas d'avoir au moins N_{low} séquences à notre disposition ? Ce peut être notamment le cas dans des applications biologiques ou médicales. Par exemple, le nombre de malades dans le cas des maladies génétiques dites orphelines est souvent assez faible. Dans le cas où les données sont issues d'expérimentations biologiques, le protocole expérimental est parfois tellement lourd qu'il ne peut être réalisé que pour un petit nombre d'individus. Paradoxalement, ces domaines requièrent des décisions très rigoureuses car les résultats influent sur des problèmes de santé publique. Les conséquences associées à des mauvaises prédictions peuvent donc être très lourdes, et il faut donc pouvoir contrôler avec précision les risques associés aux décisions prises. Or, comme nous l'avons vu, lorsque la taille de l'échantillon n'est pas suffisamment grande, les risques de trouver des faux positifs et des faux négatifs augmentent. Il faut donc trouver une solution qui nous permette de prendre en compte une plus grande quantité d'exemples relevant de la même distribution théorique. Comme nous l'avons déjà vu au chapitre 3, une solution consiste à généraliser les séquences de LS sous la forme de PDFAs. Les automates que nous manipulons étant obtenus par des procédures de fusions d'états à partir d'un PPTA, nous étudions dans la section suivante les conditions pour obtenir de bonnes fusions, et donc un bon PDFA, en vue d'une fouille de données séquentielles à partir d'automates probabilistes.

4.3 Comment apprendre un bon PDFA pour faire de la fouille de données séquentielles ?

4.3.1 Borne basse sur le nombre de symboles concernés par une fusion

En inférence grammaticale régulière, beaucoup d'articles traitent de l'apprenabilité de langages réguliers sous la forme de PDFAs (voir le chapitre 2). Tous les résultats

formels, tels que l'identification à la limite ou le cadre PAC présentés précédemment, donnent des cadres théoriques pour garantir l'apprenabilité des PDFAs à partir de séquences. Dans les deux cas, le but est de fournir les conditions sur le nombre de séquences pour apprendre le concept cible. Comme pour la plupart des résultats théoriques, ces conditions sont souvent difficiles à satisfaire dans des cas réels. Comme une condition *sur le nombre global de séquences* n'est pas toujours remplie sur les bases de données réelles, nous fournissons ici une borne basse sur le *nombre minimal de symboles* qui doivent être concernés par une fusion dans ALERGIA.

Rappelons qu'au cours du processus de fusion d'ALERGIA, la borne de HOEFFDING est utilisée pour tester la compatibilité entre deux états (voir le paragraphe 2.4.3). Le paramètre de généralisation α, utilisé dans le test, n'est autre que l'erreur de Type I d'un test d'estimation d'erreur. Nous pouvons noter que l'erreur de Type II β n'est pas utilisée dans ce test. En d'autres mots, le seul risque contrôlé par ALERGIA est le risque de rejeter à tort une bonne fusion. Dans [CO94], un résultat théorique montre néanmoins que, à la limite, les risques α et β décroissent avec l'augmentation du nombre de symboles. Cependant, rien n'est dit à propos du nombre minimal de symboles, considérés par une fusion, qui garantirait des risques donnés α et β. Nous fournissons cette borne dans cette partie.

Rappelons que deux états q_1 et q_2 sont fusionnés dans ALERGIA *ssi* :

$$\forall z \in \Sigma \cup \{\#\} \quad |\pi(q_1,z) - \pi(q_2,z)| < \sqrt{\frac{1}{2} ln(\frac{2}{\alpha})} \times \left(\frac{1}{\sqrt{n(q_1)}} + \frac{1}{\sqrt{n(q_2)}} \right),$$

où $\pi(q_1,z)$ et $\pi(q_2,z)$ sont les proportions observées, donc les estimations des vraies probabilités $p(q_1,z)$ et $p(q_2,z)$. Pour prendre en compte non seulement α mais aussi β, nous suggérons d'utiliser un test de comparaison de proportions au lieu de la borne de HOEFFDING, pour assurer la compatibilité des deux états. Comme une bonne fusion apparaît quand $\forall z$, $p(q_1,z) = p(q_2,z)$, nous testons l'hypothèse nulle $H_0 : p(q_1,z) - p(q_2,z) = 0$ contre l'hypothèse alternative $H_a : p(q_i,z) - p(q_j,z) > 0$. Notons que la direction du test dépendra des données observées. Plus précisément, pour avoir un test unilatéral, q_i sera l'état (q_1 ou q_2) à partir duquel la plus grande proportion de z sera observée, ou dit autrement $i = \arg\max_{k \in \{1,2\}} \pi(q_k,z)$.

Si $n(q_1) > 15$ et $n(q_2) > 15$, $\pi(q_i,z) - \pi(q_j,z)$ suit une distribution normale[1] :

$$\pi(q_i,z) - \pi(q_j,z) \approx \mathcal{N}\left(p(q_i,z) - p(q_j,z), \sqrt{\frac{p(q_i,z) \times \bar{p}(q_i,z)}{n(q_i)} + \frac{p(q_j,z) \times \bar{p}(q_j,z)}{n(q_j)}} \right),$$

où $\bar{p}(q_k,z) = 1 - p(q_k,z)$. Sous H_0, $p(q_1,z) = p(q_2,z) = p(q,z)$. Comme $p(q,z)$ est inconnu, nous pouvons l'estimer par $\pi(q,z)$ tel que :

$$\pi(q,z) = \frac{\pi(q_1,z) \times n(q_1) + \pi(q_2,z) \times n(q_2)}{n(q_1) + n(q_2)}. \tag{4.12}$$

[1] Pour de petits nombres de séquences, nous pouvons utiliser le test exact de FISCHER [Fis22].

Soit α l'erreur de Type I, c'est-à-dire le risque de refuser une bonne fusion. Un tel rejet surviendra quand la différence $\pi(q_i,z) - \pi(q_j,z)$ excédera une borne de rejet k, au delà de laquelle il ne reste que $\alpha\%$ de la densité de la distribution normale qui satisfait H_0. Formellement,

$$\alpha = P(H_a|H_0) = P\left(\pi(q_i,z) - \pi(q_j,z) > k|H_0\right). \tag{4.13}$$

En soustrayant la moyenne $p(q_i,z) - p(q_j,z)$ et en divisant par l'écart type $\sqrt{\frac{p(q_i,z)\times\bar{p}(q_i,z)}{n(q_i)} + \frac{p(q_j,z)\times\bar{p}(q_j,z)}{n(q_j)}}$, nous obtenons une variable Z centrée réduite qui suit une distribution normale $\mathcal{N}(0,1)$. En utilisant l'équation 4.12, nous obtenons

$$\alpha = P\left(Z > \frac{k}{\sqrt{\pi(q,z) \times \bar{\pi}(q,z)} \times \sqrt{\frac{1}{n(q_i)} + \frac{1}{n(q_j)}}}\right), \tag{4.14}$$

où $\bar{\pi}(q,z) = 1 - \pi(q,z)$. Soit β l'erreur de Type II, c'est-à-dire le risque d'accepter une mauvaise fusion. Sous H_a, $p(q_i,z) - p(q_j,z) = p_a > 0$. Nous obtenons :

$$\beta = P(H_0|H_a) = P\left(Z < \frac{k - p_a}{\sqrt{\frac{\pi(q_i,z)\times\bar{\pi}(q_i,z)}{n(q_i)} + \frac{\pi(q_j,z)\times\bar{\pi}(q_j,z)}{n(q_j)}}}\right). \tag{4.15}$$

À partir des équations 4.14 et 4.15, nous déduisons que :

$$k = z_\alpha \times \sqrt{\pi(q,z) \times \bar{\pi}(q,z)} \times \sqrt{\frac{1}{n(q_i)} + \frac{1}{n(q_j)}} \tag{4.16}$$

et

$$k = p_a - z_\beta \times \sqrt{\frac{\pi(q_i,z) \times \bar{\pi}(q_i,z)}{n(q_i)} + \frac{\pi(q_j,z) \times \bar{\pi}(q_j,z)}{n(q_j)}}, \tag{4.17}$$

où z_α (resp. z_β) correspond au $(1-\alpha)$ (resp. $(1-\beta)$) percentile de la distribution normale.

Théorème 4.2 *Pour assurer que les proportions de bonnes fusions rejetées et de mauvaises fusions acceptées n'excèdent pas respectivement des risques $\alpha_{n_{low}}$ et $\beta_{n_{low}}$, le nombre minimum de symboles n_{low} sur lequel une fusion doit être effectuée est égal à*

$$n_{low} = \frac{1+\gamma}{p_a^2} \times \left(z_{\alpha_{n_{low}}} \times \sqrt{\pi(q,z) \times \bar{\pi}(q,z)} \times \sqrt{\frac{\gamma+1}{\gamma}} + z_{\beta_{n_{low}}} \times \sqrt{\frac{(\gamma \times \pi(q_i,z) \times \bar{\pi}(q_i,z)) + \pi(q_j,z) \times \bar{\pi}(q_j,z)}{\gamma}}\right)^2,$$

où γ représente le ratio observé $\frac{n(q_j)}{n(q_i)} > 0$.

Preuve : La preuve est directe. Nous remplaçons $n(q_j)$ par $\gamma \times n(q_i)$ dans les équations 4.16 et 4.17 :

$$\begin{aligned} k &= \sqrt{\pi(q,z) \times \bar{\pi}(q,z)} \times \frac{1}{\sqrt{n(q_i)}} \times \sqrt{\frac{\gamma+1}{\gamma}} \times z_{\alpha_{n_{low}}} \\ &= p_a - z_{\beta_{n_{low}}} \times \frac{1}{\sqrt{n(q_i)}} \times \sqrt{\frac{(\gamma \times \pi(q_i,z) \times \bar{\pi}(q_i,z)) + \pi(q_j,z) \times \bar{\pi}(q_j,z)}{\gamma}} \end{aligned}$$

$\Leftrightarrow$

$$\begin{aligned} p_a = \frac{1}{\sqrt{n(q_i)}} \quad \times \quad & \left(z_{\alpha_{n_{low}}} \times \sqrt{\pi(q,z) \times \bar{\pi}(q,z)} \times \sqrt{\frac{\gamma+1}{\gamma}} \right. \\ & \left. + z_{\beta_{n_{low}}} \times \sqrt{\frac{(\gamma \times \pi(q_i,z) \times \bar{\pi}(q_i,z)) + \pi(q_j,z) \times \bar{\pi}(q_j,z)}{\gamma}} \right). \end{aligned}$$

En extrayant $n(q_i)$, nous obtenons la borne minimale $n_{low}(q_i)$ sur le nombre de symboles entrant dans l'état q_i.

$$\begin{aligned} n_{low}(q_i) = \frac{1}{p_a^2} \quad \times \quad & \left(z_{\alpha_{n_{low}}} \times \sqrt{\pi(q,z) \times \bar{\pi}(q,z)} \times \sqrt{\frac{\gamma+1}{\gamma}} \right. \\ & \left. + z_{\beta_{n_{low}}} \times \sqrt{\frac{(\gamma \times \pi(q_i,z) \times \bar{\pi}(q_i,z)) + \pi(q_j,z) \times \bar{\pi}(q_j,z)}{\gamma}} \right)^2. \end{aligned}$$

Le nombre minimal de symboles qui doivent être concernés par une fusion est alors $n_{low} = n_{low}(q_i) + n_{low}(q_j) = (1+\gamma) \times n_{low}(q_i)$, ce qui donne la borne minimale. □

4.3.2 Exemple

Nous donnons ici un exemple simple de calcul de cette borne. Si nous considérons une partie d'un PDFA représentée à la figure 4.17, supposons que nous testons la fusion entre les états 1 et 2. Calculons le nombre n_{low} de *symboles* nécessaires pour effectuer une bonne fusion.

Notons que pour chaque lettre $\{a,b,\#\}$ nous devons calculer la borne. Dans la suite, nous détaillons le cas de la lettre a et donnons directement les autres résultats. Nous avons $\pi(1,a) = \frac{70}{140}$ et $\pi(2,a) = \frac{63}{129}$. Fixons $p_a = 0.15$ et $\alpha_{n_{low}} = \beta_{n_{low}} = 5\%$, donc $z_{\alpha_{n_{low}}} = z_{\beta_{n_{low}}} = 1.64$. Le calcul de l'équation 4.12 pour la lettre a donne

$$\begin{aligned} \pi(q,a) &= \frac{\pi(1,a) \times n(1) + \pi(2,a) \times n(2)}{n(1) + n(2)} \\ &= \frac{\frac{70}{140} \times 140 + \frac{63}{129} \times 129}{140 + 129} = \frac{133}{269}. \end{aligned}$$

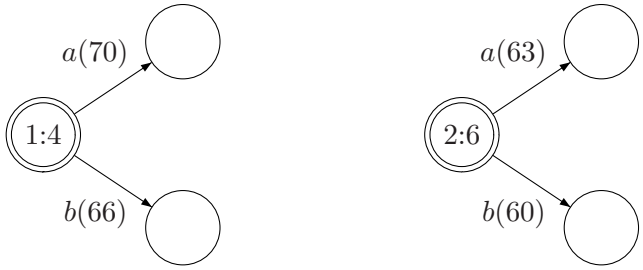

Fig. 4.17 – *Les états* 1 *et* 2 *d'un* pdfa *sont candidats à la fusion. Il y a 4 séquences qui terminent dans l'état* 1 *et 6 dans l'état* 2.

Comme $\gamma = \frac{129}{140} = 0.92$, nous obtenons :

$$\begin{aligned} n_{low} &= \frac{1+\gamma}{p_a^2} \times \left(z_\alpha \times \sqrt{\pi(q,a) \times \bar{\pi}(q,a)} \times \sqrt{\frac{\gamma+1}{\gamma}} \right. \\ &\quad \left. + z_\beta \times \sqrt{\frac{(\gamma \times \pi(1,a) \times \bar{\pi}(1,a)) + \pi(2,a) \times \bar{\pi}(2,a)}{\gamma}} \right)^2 \\ &= \frac{1+0.92}{0.15^2} \times \left(1.64 \times \sqrt{\frac{133}{269} \times \frac{136}{269}} \times \sqrt{\frac{0.92+1}{0.92}} \right. \\ &\quad \left. + 1.64 \times \sqrt{\frac{\left(0.92 \times \frac{70}{140} \times \frac{70}{140}\right) + \frac{63}{129} \times \frac{66}{129}}{0.92}} \right)^2 \\ &= 207. \end{aligned}$$

Pour b et # nous obtenons respectivement les bornes basses 207 et 79. Nous pouvons donc décider que les états 1 et 2 peuvent être fusionnés car $n(1) + n(2) = 269$, ce qui satisfait les trois bornes basses 207, 207 et 79.

4.3.3 Validation expérimentale

Pour tester l'efficacité de notre borne minimale, nous avons effectué les séries d'expériences suivantes : nous avons utilisé la grammaire de Reber [Reb67] (voir l'automate de la figure 4.12) pour échantillonner des ensembles de séquences LS_i de tailles croissantes. Pour chacun d'eux, nous apprenons deux pdfas :

- Le premier est inféré avec Alergia et la borne de Hoeffding.
- Le second est appris avec une version modifiée d'Alergia, combinant la borne de Hoeffding et notre borne minimale n_{low} (en utilisant $\alpha_{n_{low}} = \beta_{n_{low}} = 5\%$). Cela signifie que les deux bornes doivent être satisfaites pour accepter une fusion.

À partir de ces automates, nous calculons respectivement $P_H(q_0,w)$ (issu du pdfa obtenu avec la borne de Hoeffding seule) et $P_{H+n_{low}}(q_0,w)$ (issu de pdfa obtenu avec les

2 bornes) pour tous les motifs w de 1, 2 et 3 symboles, c'est-à-dire $\forall w \in (\Sigma \cup \{\lambda\})^3$ (où λ est le symbole vide), et nous les comparons avec les vraies probabilités $p(w)$ calculées à partir de la distribution cible (c'est-à-dire la grammaire de REBER). De plus, nous calculons, directement à partir des ensembles LS_i, les proportions observées $\hat{p}(w)$ de chaque motif, et nous les comparons aussi avec $p(w)$. Rappelons que $\hat{p}(w)$ est l'information utilisée par les algorithmes classiques de fouille de données séquentielles. La figure 4.18 montre les différences (moyennées et normalisées) entre les vraies probabilités et celles estimées (par les PDFAs ou par l'échantillon LS_i).

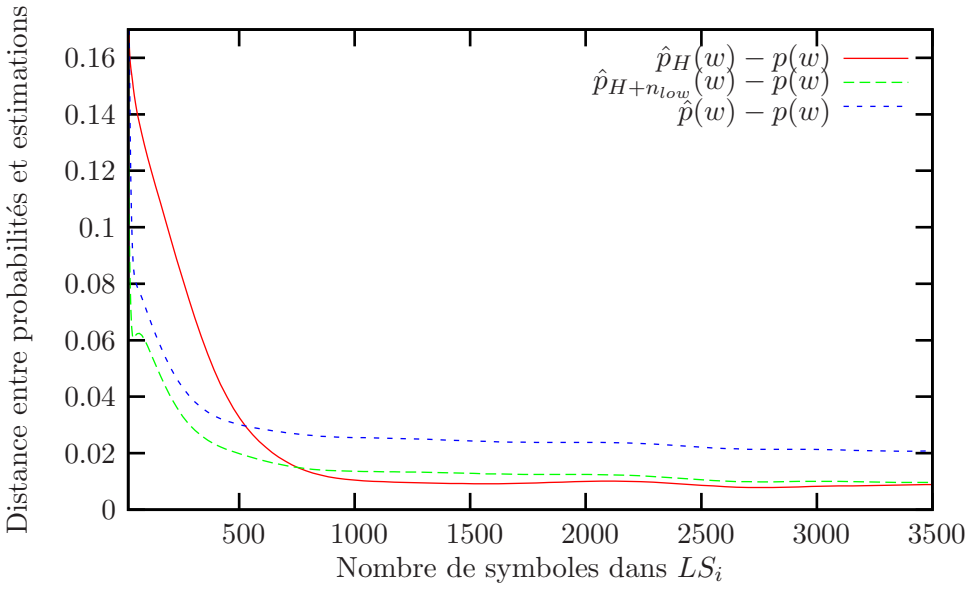

FIG. 4.18 – *Différence moyenne entre $P(q_0,w)$ et $p(w)$ sur la grammaire de* REBER.

Deux remarques importantes peuvent être faites :

- Premièrement, les deux courbes concernant l'utilisation d'un PDFA convergent vers 0 ; cela signifie qu'avec un nombre grandissant de séquences dans LS_i, le nombre de symboles concernés par la fusion d'états croit aussi, permettant de meilleures décisions durant le processus de fusion. Cependant, la borne de HOEFFDING seule n'est pas suffisante, et une utilisation conjointe avec notre borne théorique n_{low} permet d'éviter de mauvaises décisions, **particulièrement quand le nombre de symboles n'est pas très grand**. Dans ces cas, le test de notre borne permet souvent de rejeter de mauvaises fusions (donc de réduire par le risque β le taux de faux positifs) et permettant ainsi de meilleures estimations.
- Deuxièmement, la courbe tracée à partir des séquences de LS_i décroît aussi. Cependant, les estimations calculées à partir du PDFA avec la borne n_{low} sont toujours meilleures que celles calculées à partir des séquences elles-mêmes ! Cela confirme que la fouille de données séquentielles basée sur les PDFAs peut constituer une bonne alternative aux algorithmes standards, quel que soit le nombre de séquences.

4.3.4 Conclusion

Dans la première partie de ce chapitre, nous avons montré que si nous disposons d'un échantillon de taille suffisante nous pouvons garantir le contrôle des erreurs de Type I et II. Cependant, la borne minimale que nous avons proposée peut parfois être difficile à atteindre dans certains domaines.

Pour résoudre ce problème, une solution consiste à généraliser les données de l'échantillon LS sous la forme d'un modèle génératif (comme des PDFAs) dans le but de couvrir un plus grand nombre d'exemples que ceux présents dans LS. Dans cette section, nous avons présenté une borne sur le nombre de symboles concernés par une fusion permettant d'améliorer la qualité de l'automate à des fins de fouille de données séquentielles.

Afin de réduire le nombre de motifs extraits (potentiellement très grand), nous montrons dans la section suivante, qu'il est aussi possible d'inclure des contraintes probabilistes dans le processus d'extraction de motifs fréquents à partit d'un PDFA.

4.4 Nouvelles contraintes probabilistes pour la fouille de PDFAs

L'utilisation de contraintes dans les algorithmes de fouille de données séquentielles est un des enjeux actuels afin de réduire le nombre de motifs extraits. Nous avons décrit, dans le chapitre 1, comment elles sont généralement mises en place et de quelles formes elles peuvent être. Nous présentons dans cette section comment intégrer des contraintes dans l'approche de fouille de PDFAs. La première contrainte proposée permet d'extraire des motifs qui sont jugés statistiquement pertinents. Nous utilisons une fois de plus des méthodes de statistique inférentielle pour tenter d'extraire de la "vraie" connaissance. Ainsi, dans le cas où les bornes présentées précédemment ne sont pas satisfaites, cette première contrainte nous permet tout de même d'extraire de la connaissance pertinente. La seconde contrainte est directement dédiée à la réduction du nombre de motifs extraits et s'attache plus à la structure des motifs. Seuls les motifs qui apparaissent dans les séquences après un préfixe de longueur donnée sont extraits. Nous terminons cette section en présentant un nouvel algorithme de fouille de données séquentielles.

4.4.1 Pertinence d'un motif

Notre but est d'utiliser les valeurs de $P(w)$ extraites du PDFA pour vérifier la significativité statistique d'un motif fréquent. Cette significativité repose sur deux contraintes statistiques que doit respecter le motif w. La première est une contrainte de proportion, l'autre de dépendance. Les contraintes sont vérifiées à l'aide de tests statistiques.

Contrainte de proportion

Le premier test vérifie une condition *absolue*: un motif $w =< x_1 \dots x_l >$ doit couvrir une part significative de la densité de probabilité de toutes les séquences. Pour

satisfaire cette contrainte, nous appliquons un test de proportion, que nous appellerons PROP_TEST. Celui-ci a pour but de vérifier si $P(q_0,w)$ (l'estimation de $p(w)$ dans l'automate) est suffisamment grande. Pour ce faire, nous testons l'hypothèse nulle

$$H_0 \ : \ p(w) = 0,$$

contre l'hypothèse alternative

$$H_a \ : \ p(w) > 0.$$

Comme nous l'avons déjà vu, si le nombre de séquences N est suffisamment grand ($N > 30$), $P(q_0,w)$ suit asymptotiquement une distribution normale. Nous pouvons déterminer le seuil k qui définit la borne de rejet de H_0 et qui correspond au $(1 - \alpha_1)$ percentile z_{α_1} de la distribution de $p(w)$ sous H_0. Nous pouvons montrer que :

$$P\left(P(q_0,w) > k\right) = \alpha_1 \ ssi \ k = z_{\alpha_1}\sqrt{\frac{P(q_0,w)(1 - P(q_0,w))}{N}}.$$

Nous obtenons donc la règle de décision suivante : *si $P(q_0,w) > k$, la contrainte de proportion sur w est satisfaite.*

Par exemple, reprenons les séquences de la table 3.1 et considérons le motif *cc*. Dans la partie 3.2.3, nous avons montré que $P(0, < cc >) = 0.21$. En fixant $\alpha_1 = 5\%$, nous obtenons :

$$k = 1.64 * \sqrt{\frac{0.21 * (0.79)}{15}} = 0.172.$$

$P(q_0, < cc >) = 0.21 > 0.172$ donc la contrainte de proportion sur la sous-séquence $< cc >$ est satisfaite.

Contrainte de dépendance

Pour assurer le significativité d'un motif, nous lui imposons également une condition *relative* ; le but est de chercher s'il existe une dépendance entre le motif $w =< x_1 \dots x_l >$ et le motif $w^{'} =< x_1 \dots x_{l-1} >$. Plus précisément, l'objectif est de savoir si la majorité des séquences issues de la distribution cible qui contiennent $w^{'}$ contiennent aussi $w = w^{'}. < x_l >$, où "." est la fonction de concaténation. Nous avons donc deux motifs w et $w^{'}$ de probabilités $p(w)$ et $p(w^{'})$ inconnues, respectivement estimables par $\hat{p}(w)$ et $\hat{p}(w^{'})$. Pour assurer cette contrainte de dépendance, il faut que les probabilités des motifs $p(w)$ et $p(w^{'})$ soient très proches. Nous avons donc choisi d'effectuer un test de comparaison de proportions (déjà vu précédemment), que nous appellerons DEP_TEST, pour savoir si elles sont significativement différentes. Les hypothèses H_0 et H_a sont les suivantes :

$$\begin{cases} H_0 \ : \ p(w^{'}) \ = \ p(w) \ = \ p \\ H_a \ : \ p(w^{'}) \ \neq \ p(w). \end{cases}$$

Dans notre cas, l'hypothèse alternative va se réduire au cas unilatéral $p(w^{'}) > p(w)$. En effet, $w^{'}$ est une sous-séquence de w, elle aura donc toujours un support supérieur

à celui de w (voir la propriété 1.1).

$$\begin{cases} H_0 & : & p(w') & = & p(w) & = & p \\ H_a & : & p(w') & > & p(w) & & \end{cases}$$

Comme pour les tests statistiques que nous avons mis en place précédemment, les proportions suivent des lois binomiales approximables par des lois de Laplace-Gauss. Les tailles des échantillons étant identiques ($|LS| = N$), les deux lois suivent donc la même distribution :

$$\hat{p}(w),\, \hat{p}(w') \approx \mathcal{N}\left(p, \sqrt{\frac{p(1-p)}{N}}\right).$$

On a donc :

$$\hat{p}(w') - \hat{p}(w) \approx \mathcal{N}\left(0, \sqrt{p(1-p)}\sqrt{\frac{2}{N}}\right).$$

Puisque p est inconnue, on utilise son estimation :

$$\hat{p} = \frac{N \times \hat{p}(w) + N \times \hat{p}(w')}{2N} = \frac{\hat{p}(w) + \hat{p}(w')}{2}.$$

On rejettera l'hypothèse H_0 avec un risque α_2 si :

$$p(w') - p(w) > k',$$

où k' est la borne de rejet telle que

$$\begin{aligned} k' &= z_{\alpha_2} \times \sqrt{\frac{\hat{p}(w) + \hat{p}(w')}{2} \times \left(1 - \frac{\hat{p}(w) + \hat{p}(w')}{2}\right)} \times \sqrt{\frac{2}{N}} \\ &= z_{\alpha_2} \times \sqrt{\frac{(\hat{p}(w) + \hat{p}(w')) \times (2 - \hat{p}(w) - \hat{p}(w'))}{N}}. \end{aligned}$$

Nous obtenons donc la règle de décision suivante : *la contrainte de dépendance est satisfaite pour le motif w ssi $p(w') - p(w) \leq k'$, où w' est le préfixe de w.*

Prenons, par exemple, $w =< cc >$. Il faut chercher si $P(q_0, < cc >)$ et $P(q_0, < c >)$ sont statistiquement dépendants. D'après les calculs effectués au chapitre 3, $P(q_0, < cc >) = 0.21$ et $P(q_0, < c >) = 0.469$. Pour $\alpha_2 = 5\%$, on trouve $k' = 0.401$, d'où :

$$p(< c >) - p(< cc >) = 0.259 < k',$$

l'hypothèse H_0 est donc acceptée et les deux motifs sont dépendants.

En combinant les contraintes de proportion et de dépendance, nous pouvons maintenant définir ce que nous appelons un motif pertinent.

Définition 4.1 (Motif pertinent) *Un motif $w =< x_1 ... x_{l-1} x_l >$ est pertinent ssi*
(i) $P(q_0, w)$ est supérieur au seuil de support p_0,

(ii) la contrainte de proportion sur $P(q_0,w)$ est satisfaite avec un risque α_1 et
(iii) la contrainte de dépendance entre w et $w' =< x_1...x_{l-1} >$ est satisfaite avec un risque α_2.

Reprenons l'exemple ci-dessus. Avec un support $p_0 = 0.2$, α_1 et α_2 égaux à 5%, le motif $w =< cc >$ est un motif pertinent car les trois conditions sont vérifiées.

4.4.2 Contrainte de préfixe

Nous introduisons ici une deuxième contrainte qui permet de découvrir des motifs satisfaisant une longueur de préfixe donnée. Celle-ci est intéressante dans les domaines où des localisations différentes des motifs dans les chaînes peuvent exprimer des significations différentes. C'est le cas, par exemple, en bio-informatique, pour les sites de fixation des facteurs de transcription. En effet, il peut parfois être intéressant de trouver, dans une séquence d'ADN, les motifs correspondant à des situations biologiques précises (site de fixation des facteurs de transcription d'une protéine). Or, les biologistes peuvent avoir des connaissances a priori (plus ou moins claires) sur les localisations de ces motifs à l'intérieur de la séquence. Dans ce cas, indiquer une longueur de préfixe minimale permettra de se concentrer sur la recherche de ces motifs spécifiques. Comme nous le verrons dans le chapitre suivant avec notre application TRAFFICMINER, la recherche de motifs sous contrainte de préfixe peut aussi trouver son intérêt dans les applications de flux de données. Par exemple, pour connaître spécifiquement les parcours les plus empruntés en centre ville, nous aurons à imposer une contrainte de préfixe assurant que les véhicules se sont suffisamment éloignés de la périphérie urbaine en direction du centre ville.

Nous présentons dans la section suivante le calcul de la probabilité de ces motifs contraints.

Formalisation

Soit $P(S,x,\theta)$ la proportion de séquences contenant le symbole x (pas forcément le premier) à une distance θ de l'état S. Notons qu'une lettre a, à la distance 0 d'un état S, correspond à la transition sortante $q(S,a)$. Si nous reprenons l'automate de la figure 4.19, et que nous cherchons la proportion $P(0,a,2)$ de séquences contenant un a à une distance 2 de l'état de départ, nous pouvons établir que :

$$\begin{aligned} P(0,a,2) &= \pi(0,a) \times \pi(1,b) \times \pi(0,a) + \pi(0,c) \times \pi(0,c) \times \pi(0,a) \\ &\quad + \pi(0,c) \times \pi(0,b) \times \pi(2,a) + \pi(0,b) \times \pi(2,a) \times \pi(3,a) \\ &= 0.23 \times 1.0 \times 0.23 + 0.23 \times 0.23 \times 0.23 \\ &\quad + 0.23 \times 0.31 \times 1.0 + 0.31 \times 1.0 \times 0.21 \\ &= \sum_{z \in \Sigma} \pi(0,z) \times P(q(0,z),a,1) = 0.201. \end{aligned}$$

ab	bac	baba	abbac	abbaab
ba	abcc	bacc	abccc	baabba
abc	baab	ababc	babac	babaabc

Tab. 4.2 – *Ensemble de 15 séquences construit à partir de l'alphabet* $\Sigma = \{a,b,c\}$.

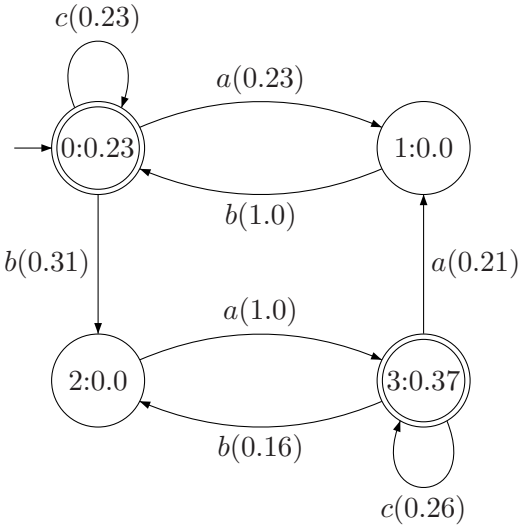

Fig. 4.19 – pdfa *inféré par* Alergia *depuis les séquences de la table 4.2.*

En observant les séquences originales de la table 4.2, nous notons que la proportion de séquences contenant a en troisième position est $\frac{3}{15} = 0.2$, donc très proche de $P(0,a,2)$.
En généralisant, nous obtenons :

$$P(S,x,\theta) = \sum_{z\in\Sigma} \pi(S,z) \times P(q(S,z),x,\theta-1) \tag{4.18}$$

$$= \sum_{T\in Q} \left(\sum_{z,q(S,z)=T} \pi(S,z) \right) \times P(T,x,\theta-1). \tag{4.19}$$

Soit $\tau_{S,T} = \sum_{z,q(S,z)=T} \pi(S,z)$ la probabilité d'utiliser une des transitions entre les états S et T. En utilisant les valeurs $\tau_{S,T}$, nous obtenons :

$$P(S,x,\theta) = \sum_{T\in Q} \tau_{S,T} \times P(T,x,\theta-1). \tag{4.20}$$

Soit $P(x,\theta)$ le vecteur des valeurs de $P(S,x,\theta)$, l'équation 4.20 devient :

$$P(x,\theta) = \tau \times P(x,\theta-1).$$

Ceci est une suite géométrique de raison τ (la matrice des valeurs $\tau_{S,T}$) et de premier terme $\pi(x)$ (le vecteur des valeurs de $\pi(S,x)$). En effet, le premier terme $P(S,x,0)$

correspond à $\pi(S,x)$, le vecteur des probabilités de transition à partir de l'état S avec la lettre x. En introduisant $\pi(x)$ dans $P(x,\theta)$, nous obtenons :

$$P(x,\theta) = \tau^{\theta} \times \pi(x). \tag{4.21}$$

Il est possible de généraliser $P(S,x,\theta)$ à $P(S,w,\theta)$, la probabilité de rencontrer un motif quelconque w à une distance θ de l'état S.

Focalisons nous tout d'abord sur le cas $w =< x_1x_2 >$, c'est-à-dire $P(S, < x_1x_2 > ,\theta)$. La position θ du motif w correspond en réalité à la position de sa première lettre. Notons qu'une séquence contenant x_1 à la position θ, suivie ensuite par x_2 peut être divisée en deux parties. La première partie contient le symbole x_1 à la position θ dans la séquence. La deuxième partie contient la suite du motif, c'est-à-dire x_2. Soit $F(S,T,x_1,\theta)$ la probabilité qu'un chemin pris au hasard, commençant à l'état S et finissant à l'état T, soit de longueur $\theta + 1$ et contienne le symbole x_1 à la θ^{ieme} position du chemin.

Nous montrons que :

$$F(S,T,x_1,\theta) = \sum_{z\in\Sigma} \pi(S,z) \times F(q(S,z),T,x_1,\theta-1) \tag{4.22}$$

$$= \sum_{R\in Q} \left(\sum_{z,q(S,z)=R} \pi(S,z) \right) \times F(R,T,x_1,\theta-1). \tag{4.23}$$

En utilisant les valeurs $\tau_{S,T}$ définies précédemment, nous avons :

$$F(S,T,x_1,\theta) = \sum_{R\in Q} \tau_{S,R} \times F(R,T,x_1,\theta-1). \tag{4.24}$$

En utilisant la notion $F(x_1,\theta)$, comme étant le vecteur des valeurs de $F(S,T,x_1,\theta)$, l'équation 4.24 devient :

$$F(x_1,\theta) = \tau \times F(x_1,\theta-1).$$

Ceci est une suite géométrique de raison τ. Le premier terme $F(x_1,0)$ correspond à la matrice de toutes les transitions avec la lettre x_1 que nous avons préalablement définie au chapitre 3.2.3 (voir l'équation 3.4). Nous obtenons :

$$F(x_1,\theta) = \tau^{\theta} \times \gamma(x_1). \tag{4.25}$$

Nous en déduisons que :

$$P(S, < x_1x_2 > ,\theta) = F(x_1,\theta) \times P(x_2). \tag{4.26}$$

Nous pouvons désormais généraliser ce principe et calculer cette probabilité pour un motif quelconque à l symboles. Nous obtenons directement :

$$P(< x_1...x_l > ,\theta) = F(x_1,\theta) \times F(x_2) \times ... \times F(x_{l-1}) \times P(x_l). \tag{4.27}$$

Rappelons que $F(S,T,x)$ (sans le paramètre θ) correspond à la probabilité qu'un chemin commençant à l'état S et terminant à l'état T contienne exactement la lettre x en dernière position.

Supposons que nous désirons calculer la probabilité du motif $< bc >$ à la distance $\theta = 1$, dans l'automate de la figure 4.19.

$$P(< bc >,1) = F(b,1) \times P(c),$$

avec

$$F(b,1) = \tau^1 \times \gamma(b).$$

Nous devons tout d'abord calculer la matrice τ ($\tau_{S,T} = \sum_{z,q(S,z)=T} \pi(S,z)$) :

$$\tau = \begin{pmatrix} 0.23 & 0.23 & 0.31 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0. & 0.21 & 0.16 & 0.26 \end{pmatrix}.$$

Il est ensuite nécessaire de calculer $\gamma(b)$, défini à l'équation 3.4, soit :

$$\gamma(b) = \begin{pmatrix} 0 & 0 & 0.31 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.16 & 0 \end{pmatrix},$$

d'où

$$F(b,1) = \tau * \gamma(b) = \begin{pmatrix} 0.23 & 0 & 0.0713 & 0 \\ 0 & 0 & 0.31 & 0 \\ 0 & 0 & 0.16 & 0 \\ 0.21 & 0 & 0.0416 & 0 \end{pmatrix}.$$

La matrice $P(c)$ a déjà été calculée à la section 3.2.2, nous obtenons donc :

$$P(< bc >,1) = \begin{pmatrix} 0.138 \\ 0.132 \\ 0.068 \\ 0.116 \end{pmatrix},$$

d'où $P(0, < bc >,1) = 0.138$, avec une proportion réelle dans la table 4.2 de $\frac{3}{15}$.

Relaxation de la contrainte de préfixe

Sans remettre en cause l'intérêt de notre contrainte de préfixe, on peut constater qu'il sera parfois difficile dans certaines applications de déterminer exactement la valeur pertinente de θ. Puisque nous avons cité l'intérêt d'une telle approche en biologie moléculaire, il faudrait pouvoir tenir compte d'éventuelles mutations génétiques pouvant entraîner l'insertion ou la délétion de certains symboles dans la séquence. Dans ce cas, la recherche d'un motif à une position fixée peut être inappropriée. Dans le but de relâcher la contrainte, nous introduisons une variable ϵ qui permet de découvrir des motifs à une position $\theta \pm \epsilon$. C'est-à-dire que le motif recherché pourra être positionné entre les

lettres à la position $\theta-\epsilon$ et $\theta+\epsilon$, avec ϵ un entier compris entre 0 et θ. Pour calculer la probabilité d'un motif à la distance $\theta \pm \epsilon$, il suffit de prendre en compte les probabilités du motif à toutes les positions possibles. En utilisant les formules précédentes, nous pouvons facilement montrer que :

$$\begin{aligned} P(< x_1...x_l >,\theta \pm \epsilon) &= P(< x_1...x_l >,\theta) \\ &+ \sum_{i=1}^{\epsilon} (P(< x_1...x_l >,\theta+i) + P(< x_1...x_l >,\theta-i)). \end{aligned} \tag{4.28}$$

Nous pouvons donc définir désormais notre contrainte de préfixe.

Définition 4.2 (Contrainte de préfixe) *Un motif* $w =< x_1...x_{l-1}x_l >$ *vérifie la contrainte de préfixe, pour des valeurs données de* $\theta \geq 0$ *et* $0 \leq \epsilon \leq \theta$ *ssi* $P(w,\theta \pm \epsilon)$ *est supérieur au seuil de support* p_0.

4.4.3 Caractéristiques d'anti-monotonie des contraintes

Nous avons présenté, dans la section 1.4.1, les définitions des contraintes monotones et anti-monotones. Pour utiliser les techniques d'élagage et la méthode de construction des candidats mise en place dans les algorithmes type APRIORI, il faut que les contraintes soient anti-monotones.

Dans cette section, nous montrons que cette condition n'est remplie que par certaines de nos contraintes.

Contrainte de pertinence

Rappelons que la contrainte de pertinence est vérifiée si les trois conditions suivantes sont vraies :

(i) $P(q_0,w)$ est supérieur au seuil de support p_0,
(ii) la contrainte de proportion sur $P(q_0,w)$ est satisfaite avec un risque α_1 et
(iii) la contrainte de dépendance entre w et $w' =< x_1...x_{l-1} >$ est satisfaite avec un risque α_2.

Il faut donc étudier le caractère anti-monotone de chacune de ces trois conditions :

(i) la contrainte de support est clairement anti-monotone d'après la propriété 1.1. Cette propriété reste vérifiée si le calcul du support se fait à partir de l'automate. En effet, dans le PDFA, les chemins d'acceptation de la sous-séquence $w^{'}$ sont forcément inclus dans ceux de la séquence w.

(ii) Concernant la contrainte de proportion, il faut donc montrer que si w vérifie le test, c'est-à-dire $P(q_0,w) > k$ avec $k = z_{\alpha_1} * \sqrt{\frac{P(q_0,w)*(1-P(q_0,w))}{N}}$, alors toute sous-séquence $w^{'}$ la vérifie aussi, c'est-à-dire $P(q_0,w^{'}) > k$ avec $k = z_{\alpha_1} * \sqrt{\frac{P(q_0,w')*(1-P(q_0,w'))}{N}}$. Intuitivement, la contrainte parait anti-monotone, puisque

la probabilité d'un motif w est toujours supérieure ou égale à la probabilité de tous ces sous-motifs w'. Cependant la borne k dépend aussi de la probabilité du motif concerné. Nous allons donc étudier le comportement de la fonction

$$h(p) = p - z_{\alpha_1} * \sqrt{\frac{p * (1-p)}{N}},$$

où, par souci de simplification de notation, p est la proportion d'un motif. Cette fonction est représentée à la figure 4.20 pour certaines valeurs de z_{α_1} et N, même si la forme de la courbe reste toujours la même quelles que soient ces valeurs. Si la séquence w vérifie le test de proportion, cela signifie que $h(P(q_0,w))$ est strictement positive et on peut observer alors que l'on se situe sur la partie monotone croissante de la courbe. Comme toutes les sous-séquences w' ont une probabilité supérieure à $P(q_0,w)$ avec les mêmes N et α_1, $h(P(q_0,w')$ sera toujours positive et on aura donc toujours $P(q_0,w') > k$. **La contrainte de proportion est donc anti-monotone**.

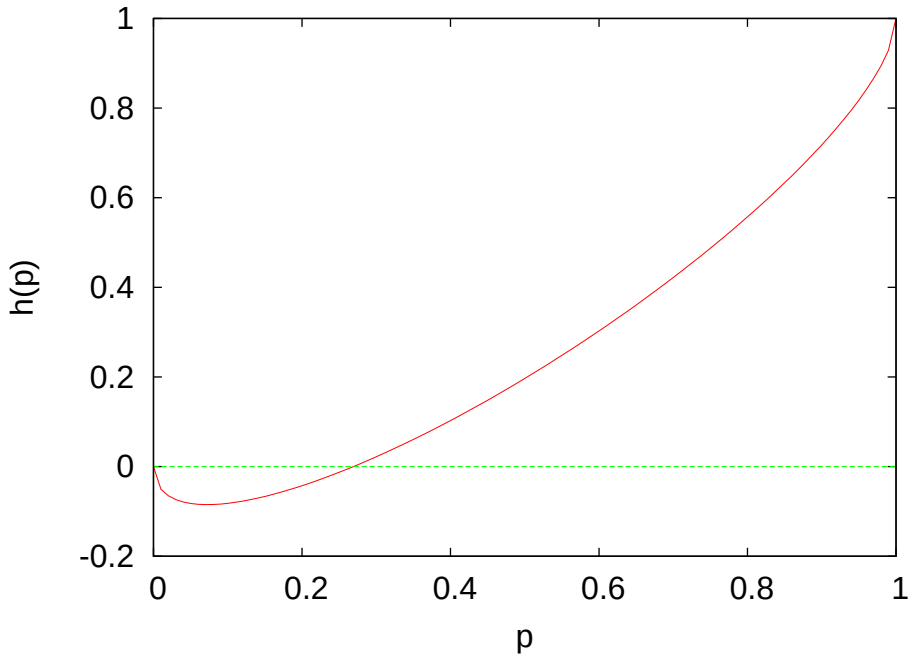

Fig. 4.20 – *Évolution de la courbe $h(p)$ en fonction de la probabilité p, pour $N = 15$ et $z_{\alpha_1} = 1.64$ (soit $\alpha_1 = 5\%$).*

(iii) il faut enfin monter que si un motif vérifie la contrainte de dépendance tous ses sous-motifs la vérifient aussi. Or, intuitivement cette affirmation semble incorrecte. En effet, prenons par exemple un motif $< abc >$ vérifiant la contrainte de dépendance. Cela signifie que, dans l'automate et dans les données séquentielles,

w	$P(w)$
abc	0.5
ab	0.55
ac	0.6
a	0.8

TAB. 4.3 – *Probabilités de divers motifs.*

on trouve "souvent" un c après le motif $< ab >$. Maintenant, si nous considérons le sous-motif $< ac >$ et supposons qu'il ne vérifie pas la contrainte, cela n'empêche pas que $< ab >$ et $< abc >$ puissent la vérifier. Donnons un contre-exemple montrant que la contrainte n'est pas anti-monotone. Supposons que nous disposons d'un échantillon de $N = 1000$ séquences avec des motifs dont les probabilités sont résumées dans la table 4.3. Testons la dépendance entre $< abc >$ et $< ab >$, avec un risque $\alpha_2 = 2.5\%$, soit $z_{\alpha_2} = 1.96$. $k^{'}$ vaut 0.06, $p(< ab >) - p(< abc >)$ vaut 0.05 ; donc la contrainte de dépendance est satisfaite pour le motif $< abc >$ (car $p(< ab >) - p(< abc >) < k^{'}$). Maintenant, considérons les motifs $< ac >$ et $< a >$. $k^{'}$ vaut 0.056, $p(< a >) - p(< ac >)$ vaut 0.2 ; donc la contrainte de dépendance n'est pas satisfaite pour le motif $< ac >$ (car $p(< ab >) - p(< abc >) > k^{'}$). **La contrainte de dépendance n'est donc pas anti-monotone**. Pour contourner ce petit inconvénient, nous présenterons dans la suite la méthode que nous avons mise en place pour construire efficacement les motifs candidats.

Contrainte de préfixe

Rappelons que la contrainte de préfixe est vérifiée, pour un motif w, une longueur de préfixe θ, une valeur de relaxation ϵ et un seuil de support p_0 *ssi*:

$$P(w,\theta \pm \epsilon) \geq p_0.$$

Pour assurer la condition d'anti-monotonie, il suffit de montrer que

$$\forall\ w^{'}\ tq\ w^{'} \prec w,\ P(w^{'},\theta) \geq p_0.$$

En effet, l'ajout du paramètre ϵ n'a pas d'influence sur le caratère monotone de la contrainte.

Cette condition n'est pas vraie si l'on considère la définition d'inclusion telle qu'elle a été présentée à la définition 1.4. En effet, prenons un motif abc vérifiant la contrainte pour $\theta = 2$, cela signifie que la lettre a est à une distance 2 de l'état initial et que l'on trouve ensuite les lettres b et c sans considérations particulières de distance. Donc rien ne permet de conclure que la contrainte sera vérifiée pour le motif bc à une distance 2. Cette contrainte n'est donc pas anti-monotone pour la relation d'inclusion de la définition 1.4. Néanmoins, si une séquence $w =< x_1 \dots x_l >$ vérifie la contrainte pour une longueur de préfixe θ alors **tout préfixe de cette séquence** vérifiera la contrainte

pour la même longueur de préfixe. En effet, tous les chemins dans l'automate contenant le motif $< x_1 \dots x_l >$ à la distance θ contiennent aussi les sous motifs commençant par x_1. Nous utiliserons donc cette propriété dans la construction des motifs candidats.

4.4.4 Algorithme ACSM

En combinant les contraintes que nous avons mises en place, nous sommes désormais en mesure de proposer un nouvel algorithme de fouille de données séquentielles. L'objectif est de découvrir, à partir d'un PDFA, tous les motifs fréquents et pertinents, en fonction d'un seuil de support p_0 et de deux risques statistiques α_1 et α_2. Le pseudo-code de notre algorithme ACSM (Automata-based Constrained Sequence Mining) est présenté dans l'algorithme 4.1. Cette première version ne tient pas compte de la contrainte de préfixe (nous l'aborderons plus tard). Dans une première partie, des lignes 4.0.2 à 4.0.9, ACSM initialise un ensemble de motifs fréquents et pertinents composés d'un unique symbole. Comme aucun motif n'a encore été extrait, seuls les tests du support (ligne 4.0.4) et de proportion (PROP_TEST ligne 4.0.5) sont effectués. A chaque étape n, les motifs fréquents de longueur n sont conservés dans l'ensemble G_n. Les chemins de longueur 1 du PDFA qui ne satisfont pas ces deux tests ne seront plus étudiés par la suite, ce qui permet un premier élagage de l'espace de recherche. La seconde partie de ACSM permet de rechercher les motifs de longueurs supérieures à 2 en utilisant les motifs de longueurs inférieures. Pour qu'un nouveau motif soit accepté, il faut que trois conditions soient satisfaites :

1. le test du support (ligne 4.0.17),
2. le test de proportion Prop_test (ligne 4.0.18) et
3. le test de dépendance Dep_test (ligne 4.0.19).

Nous avons montré ci-dessus que la contrainte de dépendance n'était pas anti-monotone. Il est donc nécessaire de conserver les motifs qui ont été refusés uniquement par le test de dépendance. Nous utilisons donc l'ensemble G'_n qui contient tous les motifs acceptés à l'étape n et les motifs qui ont passé tous les tests sauf celui de dépendance. La génération des candidats de l'étape $n+1$ se fait donc grâce à la fonction CONSTRUCTION_CANDIDATS (ligne 4.0.16), inspirée des fonctions de génération type Apriori. Cette fonction est détaillée dans l'algorithme 4.2.

Un candidat c de longueur n y est construit si le sous-motif constitué de ces $n-1$ premières lettres et celui constitué des $n-1$ dernières appartiennent à G'_n. Il est ensuite ajouté à la liste des candidats si tous ces sous-motifs appartiennent aussi à G'_n. Notons qu'il est uniquement nécessaire de tester les sous-motifs de longueur n. À la ligne 4.1.6 de l'algorithme 4.2, le point désigne l'opération de concaténation.

Nous présentons une deuxième version de l'algorithme (voir algorithme 4.3) utilisant la contrainte de longueur de préfixe. Dans ce cas, la fonction CONSTRUCTION_CANDIDATS n'est plus appropriée. En effet, nous avons montré que la contrainte de longueur de préfixe n'était pas anti-monotone. Pour construire les candidats, nous utilisons donc le fait que la propriété soit anti-monotone sur les préfixes. Il suffit donc d'utiliser les motifs fréquents de l'étape précédente et de leur ajouter les lettres de l'alphabet (modification

Algorithme 4.1 : Pseudo-code de ACSM.

Entrées : Un PDFA A = (Q, Σ, q, q_0, π, π_F), un seuil de support p_0, deux risques α_1 et α_2

Sorties : Un ensemble G de motifs fréquents pertinents

4.0.1 **début**
4.0.2 $G_1 \leftarrow \emptyset$;
4.0.3 **pour chaque** $l \in \Sigma$ **faire**
4.0.4 **si** $P(q_0,l) \geq p_0$ **alors**
4.0.5 **si** PROP_TEST *($P(q_0,l)$,α_1) est vérifié* **alors**
4.0.6 $G_1 \leftarrow G_1 \cup \{l\}$;
4.0.7 **fin**
4.0.8 **fin**
4.0.9 **fin**
4.0.10 $G \leftarrow G_1$;
4.0.11 $n \leftarrow 1$;
4.0.12 $G_1^{'} = G_1$;
4.0.13 **tant que** $G_n^{'} \neq \emptyset$ **faire**
4.0.14 $G_{n+1} \leftarrow \emptyset$;
4.0.15 $C_{n+1} \leftarrow \emptyset$;
4.0.16 **pour chaque** $v \in$ CONSTRUCTION_CANDIDATS *($G_n^{'}$)* **faire**
4.0.17 **si** $P(q_0,v) \geq p_0$ **alors**
4.0.18 **si** PROP_TEST *($P(q_0,v)$,α_1) est vérifié* **alors**
4.0.19 **si** DEP_TEST ($\overrightarrow{V_{in}}$,$\overrightarrow{V_{out}}$,α_2) *est vérifié* **alors**
4.0.20 $G_{n+1} \leftarrow G_{n+1} \cup \{v\}$;
4.0.21 **sinon**
4.0.22 $G_{n+1}^{'} \leftarrow G_{n+1}^{'} \cup \{v\}$;
4.0.23 **fin**
4.0.24 **fin**
4.0.25 **fin**
4.0.26 **fin**
4.0.27 $G_{n+1}^{'} \leftarrow G_{n+1}^{'} \cup G_{n+1}$;
4.0.28 $G \leftarrow G \cup G_{n+1}$;
4.0.29 $n \leftarrow n+1$;
4.0.30 **fin**
4.0.31 **retourner** G ;
4.0.32 **fin**

Algorithme 4.2 : Algorithme CONSTRUCTION_CANDIDATS.

Entrées : Un ensemble $G_n^{'}$ contenant des motifs de longueur n
Sorties : Un ensemble C_{n+1} de motifs candidats de longueur $n+1$

```
4.1.1  début
4.1.2      C_{n+1} ← ∅ ;
4.1.3      pour tous les v ∈ G'_n faire
4.1.4          pour tous les w ∈ G'_n faire
4.1.5              si pour tout i de 1 à n − 1, v_i = w_{i+1} alors
4.1.6                  c = v. < w_n >;
4.1.7                  si toutes sous-séquences de c ∈ G'_n alors
4.1.8                      C_{n+1} ← c ;
4.1.9      retourner C_{n+1};
4.1.10 fin
```

des lignes 4.0.15 et 4.0.16). De plus, il est nécessaire d'inclure les nouveaux paramètres θ et ϵ, et de remplacer tous les calculs de $P(q_0,v)$ par $P(q_0,v,\theta \pm \epsilon)$.

4.4.5 Résultats expérimentaux

Nous allons, dans cette section, étudier l'intérêt des contraintes que nous avons proposées dans la section précédente. L'objectif attendu lors de la mise en place de contraintes dans les algorithmes de fouille de données est double. Premièrement, on souhaite diminuer l'espace de recherche et donc obtenir un ensemble de motifs plus restreint. Deuxièmement, par la réduction du nombre de motifs, on souhaite pouvoir plus facilement les étudier et répondre ainsi à des besoins spécifiques, généralement dépendants de l'application.

Nous allons donc évaluer les effets individuels de chacune des contraintes sur le nombre de motifs extraits. Pour cette expérimentation nous avons utilisé un programme de génération (IBM DATAGEN) très utilisé en fouille de données[2]. Nous avons donc généré une base, que nous appellerons SYNT, de 10000 séquences composées chacune d'environ 10 itemsets. En utilisant l'algorithme ACSM, nous avons évalué individuellement l'effet de chacune des contraintes sur le nombre de motifs extraits.

Sur les figures 4.21 et 4.22, nous n'utilisons pas la contrainte de longueur de préfixe. Ces deux premiers ensembles de courbes ont été calculés sur la base SYNT, et montrent seulement l'effet des contraintes de pertinence. L'influence du test de proportion sans le test de dépendance est présenté en figure 4.21, et l'impact seul du test de dépendance est montré à la figure 4.22. Logiquement, nous pouvons observer que plus les contraintes sont fortes et plus le nombre de motifs extraits diminue. De plus, nous pouvons noter que plus le seuil de support p_0 augmente et plus les contraintes de pertinence deviennent inutiles. Les courbes de la figure 4.23 montrent l'influence de la contrainte de longueur

2 http://www.cs.rpi.edu/∼zaki/software/

Algorithme 4.3 : Pseudo-code de ACSM avec la contrainte de longueur de préfixe.

Entrées : Un PDFA A = (Q, Σ, q, q_0, π, π_F), un seuil de support p_0, deux risques α_1 et α_2, une longueur de préfixe θ et un paramètre de relaxation ϵ

Sorties : Un ensemble G de motifs fréquents pertinents

```
4.2.1  début
4.2.2      G_1 ← ∅ ;
4.2.3      pour chaque l ∈ Σ faire
4.2.4          si P(q_0,l,θ ± ε) ≥ p_0 alors
4.2.5              si PROP_TEST (P(q_0,l,θ ± ε),α_1) est vérifié alors
4.2.6                  G_1 ← G_1 ∪ {l} ;
4.2.7              fin
4.2.8          fin
4.2.9      fin
4.2.10     G ← G_1 ;
4.2.11     n ← 1 ;
4.2.12     tant que G_n ≠ ∅ faire
4.2.13         G_{n+1} ← ∅ ;
4.2.14         pour chaque w =< x_1...x_n >∈ G_n faire
4.2.15             pour chaque x' ∈ Σ faire
4.2.16                 v ← w. < x' > ;
4.2.17                 si P(q_0,v,θ ± ε) ≥ p_0 alors
4.2.18                     si PROP_TEST (P(q_0,v,θ ± ε),α_1) est vérifié alors
4.2.19                         si DEP_TEST (V_in→,V_out→,α_2) est vérifié alors
4.2.20                             G_{n+1} ← G_{n+1} ∪ {v} ;
4.2.21                         fin
4.2.22                     fin
4.2.23                 fin
4.2.24             fin
4.2.25         fin
4.2.26         G ← G ∪ G_{n+1} ;
4.2.27         n ← n + 1 ;
4.2.28     fin
4.2.29     retourner G ;
4.2.30 fin
```

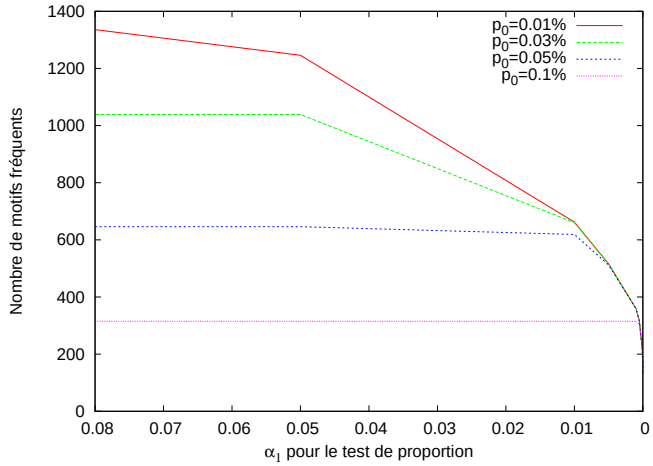

Fig. 4.21 – *Effet de la contrainte de proportion sur le nombre de motifs extraits.*

de préfixe θ. Nous pouvons observer que plus la taille du préfixe augmente, c'est-à-dire plus la contrainte est forte (pour une configuration donnée de p_0, α_1 et α_2) et plus le nombre de motifs extraits décroît. Il est intéressant également de remarquer que l'effet de θ est très rapide et que le nombre de motifs décroît très vite.

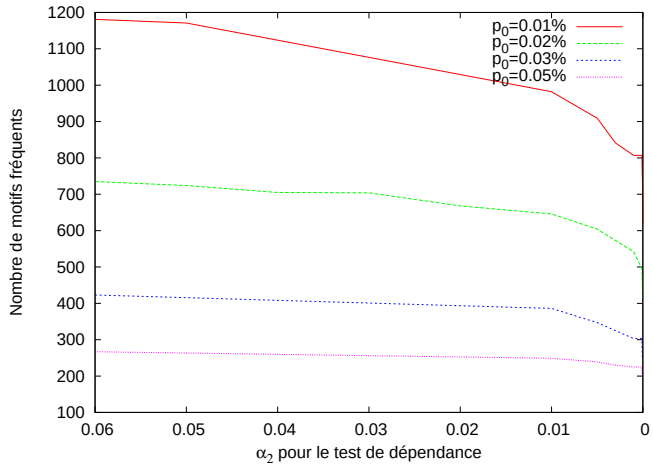

Fig. 4.22 – *Effet de la contrainte de dépendance sur le nombre de motifs extraits.*

4.5 Conclusion

Nous avons montré dans cette troisième contribution que l'utilisation des PDFAs pour faire de la fouille de données séquentielles n'était pas un obstacle à l'intégration de contraintes autres que la contrainte de fréquence. En effet, nous avons pu introduire

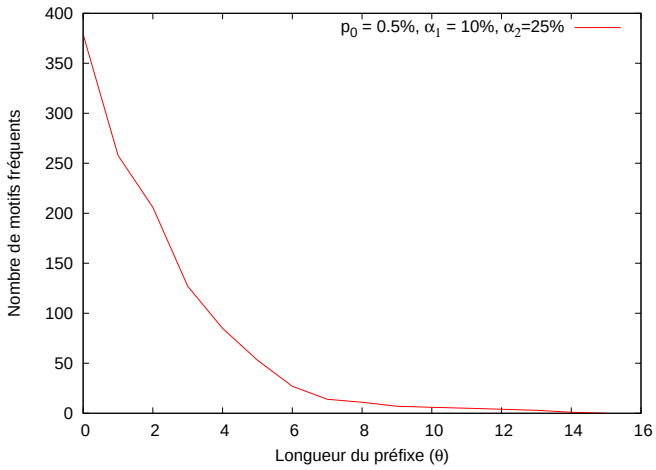

FIG. 4.23 – *Effet de la contrainte de longueur de préfixe sur le nombre de motifs extraits.*

une contrainte de pertinence permettant d'extraire des motifs non seulement fréquents mais aussi statistiquement pertinents. Cette contrainte paraît particulièrement utile lorsque les bornes présentées précédemment ne sont pas respectées. Nous avons aussi introduit une contrainte structurelle sur la forme des motifs.

Nous allons maintenant aborder la mise en œuvre de notre algorithme dans un cas pratique. Nous présentons dans le chapitre suivant une application liée au trafic routier et nous mettons en avant un des problèmes présentés dans le chapitre 1, la préservation de la vie privée. En effet, nous montrons en quoi l'approche basée sur les PDFAs peut, dans certains cas particuliers, être une solution à ce problème.

5 Fouille de PDFAs et préservation de la vie privée

Sommaire

Résumé

Jusqu'à présent nous avons supposé que nous disposions d'un ensemble de séquences LS à partir desquelles un algorithme d'inférence grammaticale était capable d'apprendre un PDFA permettant de modéliser un langage régulier. L'intérêt d'une telle approche est de disposer d'une représentation compacte et compréhensible permettant de représenter potentiellement une quantité de séquences plus importante que dans LS et sur laquelle il est possible d'effectuer une tâche de fouille de données séquentielles. Un autre avantage important de cette représentation basée sur les PDFAs est son exploitation lorsque nous n'avons pas directement accès à un ensemble de séquences bien qu'elles soient présentes de façon sous-jacentes dans le problème. Nous montrons dans ce chapitre que la question de la préservation de la vie privée peut être appréhendée au travers des PDFAs : nous montrons une application de trafic urbain justifiant l'intérêt de cette approche.

5.1 Fouille de données séquentielles préservant la vie privée

5.1.1 Introduction

Comme nous l'avons mentionné dans la section 1.5, le respect de la vie privée est un nouvel enjeu important dans le cadre de la fouille de données.

Dans le but de traiter un ensemble de séquences tout en prenant soin de ne pas porter atteinte à la vie privée des individus qui leurs sont associés, nous proposons d'utiliser

les techniques de fouille de données basées sur l'utilisation de pdfas présentées dans les sections précédentes.

Une première approche naïve peut consister à proposer d'utiliser les séquences d'origine, pour inférer un pdfa, puis à s'engager à détruire les séquences dès le pdfa construit. Cette solution n'est pas satisfaisante pour les individus concernés car rien ne permet d'assurer que les séquences seront effectivement détruites.

Il existe, par contre, une classe d'applications où notre approche se trouve naturellement être très pertinente dans le cadre de la préservation de la vie privée. Il s'agit des applications mettant en œuvre des flux dans des graphes. Deux applications typiques illustrant ce domaine sont la fouille des usages du Web et la gestion de flux routiers dans une ville.

La fouille du Web est classiquement étudiée sous trois aspects principaux : la fouille de la structure du Web, la fouille du contenu du Web et la fouille des usages du Web [KB00]. La fouille de données séquentielles est particulièrement bien adaptée à cette dernière famille d'applications au Web. En effet, dans le cadre de la fouille des usages du Web, l'objectif est de découvrir des modèles de façons de parcourir un site Web donné par les internautes. Il s'agit ici en fait de découvrir les séquences de pages Web fréquemment observées par les visiteurs du site. L'objectif final affiché est de pouvoir adapter dynamiquement un site Web à chaque visiteur en fonction de son comportement observé sur un certain échantillon du site qu'il a commencé à visiter. Cette thématique a été abordée dès 1997 dans le cadre des recherches sur les sites Web adaptatifs [PE97] et a connu depuis un essor considérable [Mob06]. Les techniques couramment utilisées se fondent sur l'utilisation des fichiers logs associés à chaque site Web. A chaque fois qu'un visiteur observe une page d'un site Web, un certain nombre d'informations le concernant sont conservées sur le serveur du site : numéro IP, nom de la page, heure de chargement, taille du chargement, etc. Il est évident qu'une telle pratique porte atteinte à la vie privée des visiteurs. Même si des techniques, comme celle du P3P [RC99], permettant de définir des politiques de préférences sur la confidentialité, sont couramment utilisées sur de nombreux sites Web de nos jours, le visiteur d'un site Web n'est absolument pas certain du respect de ces politiques par les serveurs des sites Web. De plus, la présence de proxy cache au sein des réseaux rend très difficile l'utilisation des fichiers logs [Mur06]. En effet, de tels dispositifs conduisent en permanence à générer des fichiers logs ne contenant pas toutes les informations sur toutes les pages téléchargées et les numéros IP des internautes étant parfois ceux de proxy, il devient délicat d'associer correctement un enregistrement donné du fichier à un internaute donné. De nombreuses techniques de pré-traitement ont été proposées depuis plusieurs années afin de tenter de reconstruire les logs mais les résultats ne sont jamais totalement satisfaisants et il est évident que fouiller des séquences de pages visitées de mauvaise qualité ne peut conduire qu'à l'extraction de connaissances non pertinentes. Les techniques que nous avons mises en œuvre permettent de proposer une alternative fiable aux techniques de fouille basées sur les fichiers logs en proposant une nouvelle vision de la fouille n'utilisant plus aucune donnée historique reliée aux utilisateurs. La structure du site Web constitue la base de l'automate qu'il faudra fouiller. Chaque page constitue un état de l'automate et chaque hyperlien est une transition de l'automate. La probabilité assignée à chaque transition

correspond au pourcentage d'utilisateurs ayant cliqué sur le lien correspondant, par rapport au nombre d'utilisateurs entrant sur la page. La probabilité de chaque état (probabilité d'être final) est le pourcentage d'utilisateurs qui ont quitté le site par cette page. En fait, grâce à notre technique, seul le comptage des utilisateurs sur les liens et les pages permet d'extraire les chemins fréquemment empruntés par les internautes visitant un site Web, aucune information d'ordre privé n'est plus requise, préservant ainsi la vie privée des internautes.

La gestion de flux routier dans une ville est un second domaine d'application intéressant pour notre système. De nombreuses recherches existent dans ce secteur et le respect de la vie privée y est un problème central. De façon simple, on pourrait en effet, suivre les chemins empruntés par des véhicules en utilisant par exemple des caméras vidéo. Disposant des séquences de rues empruntées par un ensemble de conducteurs, il serait alors possible d'utiliser un algorithme classique de fouille de séquences pour y extraire des ensembles de séquences fréquentes. Une telle approche serait cependant une atteinte insupportable à la vie privée des conducteurs qui s'opposeraient violemment à une telle méthode.

Divers travaux ont cherché à emprunter d'autres voies plus réalistes. Nous avons, par exemple, déjà évoqué les travaux de Gidofalvi et al. [GHP07] qui recherchent des routes fréquemment utilisées avec un soucis de préservation de la vie privée. Dans un cadre quelque peu différent, dans [LPFK07], les auteurs utilisent les données issues d'un gps personnel pour créer un système d'assistance à la navigation personnelle. Un HMM est construit à partir de ces données et celui-ci est ensuite utilisé pour aider l'utilisateur à prendre des bonnes décisions concernant le bon chemin, le bus adéquat, etc. Le système est capable de prédire la destination de l'individu. Dans cette approche, les notions de préservation de la vie privée ne sont pas réellement abordées.

Une approche basée sur l'utilisation des automates peut permettre d'échapper à cet inconvénient. En effet, si l'on considère une carte routière, les croisements peuvent constituer les états simples (non finaux et non initiaux). Les états initiaux et finaux représentent respectivement les portes d'entrée et de sortie de la carte. Les transitions modélisent quant à elles les routes. Les probabilités sont obtenues en utilisant des compteurs sur les transitions et les états finaux (les autres états ont un compteur nul). Pour chaque voiture, nous n'avons donc plus besoin de connaître la plaque d'immatriculation, ou encore les rues qu'elle traverse. Il n'est donc plus nécessaire de disposer d'outils de suivi des voitures. Malgré le fait que nous ne disposons pas des routes individuelles empruntées par chaque conducteur, nous sommes capables, avec acsm, d'extraire des chemins fréquents et pertinents, c'est-à-dire des successions de rues non nécessairement consécutives, empruntées par les véhicules au sein de la ville.

Dans le contexte des problèmes de contrôle de flux, il est donc évident que de nombreuses autres applications peuvent être abordées à la lumière de notre approche, par exemple la gestion de flux de voyageurs dans un aéroport en vue de l'installation de tapis roulant, la gestion des flux d'objets sur une chaîne de production en vue de renforcer les capacités des machines positionnées sur les flux les plus importants, ou également la gestion des chemins empruntés par des clients dans des hypermarchés afin d'optimiser les placements des produits pour augmenter la rentabilité des magasins

tout en préservant le bien être des clients.

Un problème important se pose toutefois dans le cas où l'on cherche à utiliser notre approche avec un automate construit non pas par inférence à l'aide d'un ensemble de séquences, mais directement à l'aide des données de flux comme dans le cas des applications présentées ci-dessus. En effet, nous avons vu qu'il existe des conditions théoriques permettant d'assurer la pertinence d'un automate lorsqu'il est appris à partir d'un ensemble de séquences. Dans le cas où l'automate est construit sans utiliser de séquences, nous ne pouvons plus assurer sa pertinence de cette façon. Dans la suite de cette section, nous proposons d'utiliser la longueur des séquences sous-jacentes à l'automate comme critère de qualité de celui-ci. Ne disposant bien sûr pas de ces séquences, nous allons montrer comment il est possible de calculer l'espérance de la taille de celle-ci de façon formelle.

5.1.2 Étude sur la longueur des séquences

Il est connu, en inférence grammaticale, que la longueur des séquences d'entrée a un impact direct sur la convergence des algorithmes d'inférence. Cela peut être expliqué par le fait que les pdfas ont des difficultés à modéliser des dépendances à long terme (voir [Cal07]) car ils sont basés sur les propriétés de Markov. Sans aucune information sur le nombre de séquences nécessaires à la construction de l'automate, nous montrons dans cette partie que nous sommes capables de calculer l'espérance de la longueur des séquences modélisées par un pdfa. Nous montrons qu'elle donne une bonne information sur la qualité de cet automate. Commençons dans un premier temps par étudier expérimentalement l'influence de la longueur des séquences sur la qualité des automates inférés.

Étude expérimentale

Nous avons échantillonné plusieurs ensembles LS_i de séquences de longueurs différentes (5, 10 et 30 caractères). Ensuite, nous avons inféré les automates correspondants avec l'algorithme Alergia. Pour chacun des ensembles, nous avons calculé les probabilités de tous les trigrammes, bigrammes et unigrammes sur l'ensemble d'apprentissage et sur l'automate appris. Nous avons ensuite calculé les différences entre les probabilités calculées sur LS_i ($p(w)$) et celles estimées à l'aide de l'automate ($P(q_0,w)$). Les différences sont sommées puis le tout est normalisé par le nombre total de motifs considérés. Nous présentons les résultats à la figure 5.1 où nous pouvons noter que plus la longueur des séquences est petite et plus la différence est faible et donc meilleures sont les estimations données par le pdfa. Dans ce contexte, il paraît intéressant de pouvoir estimer l'espérance de la longueur des séquences pour estimer ainsi la qualité des pdfas.

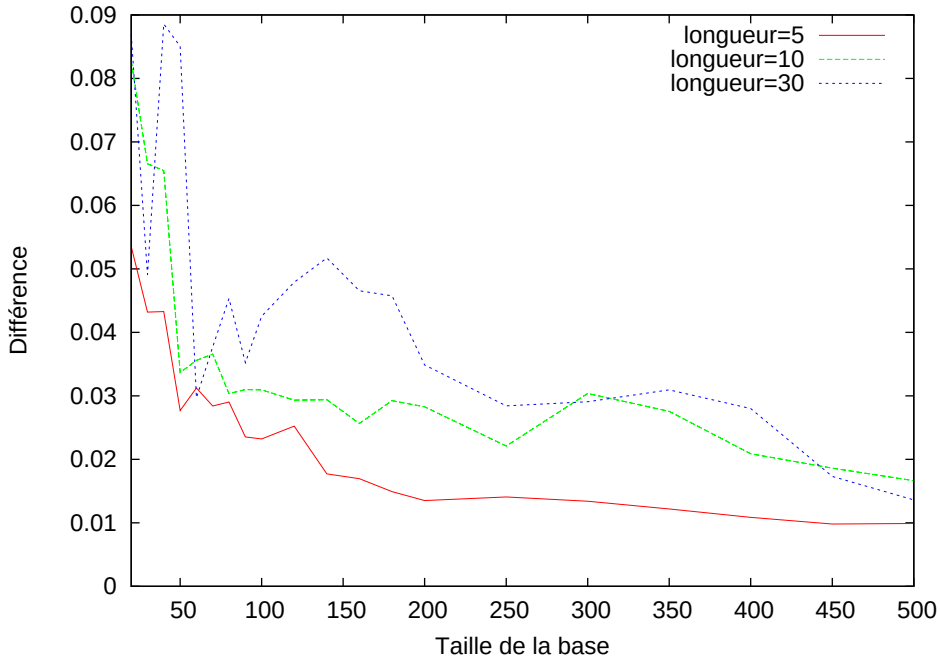

FIG. 5.1 – *Différence moyenne entre $P(q_0,w)$ et $p(w)$.*

Espérance de la longueur des séquences

Soit l la longueur d'une séquence acceptée par un PDFA (c'est-à-dire terminant dans un état final). l est une variable aléatoire dont l'espérance mathématique est égale à :

$$E(l) = \sum_{\delta=0}^{\infty} \delta \times p(l = \delta), \tag{5.1}$$

où $p(l = \delta)$ est la probabilité qu'une séquence possède δ lettres. Comme, dans le cas qui nous intéresse, le PDFA ne résulte pas d'une phase d'inférence à partir des séquences, $p(l = \delta)$ est inconnu. Cependant, il peut être estimé par $P(q_0,l = \delta)$. Dans la section 4.4.2, nous avons défini $\tau_{S,T} = \sum_{z,q(S,z)=T} \pi(S,z)$ comme la probabilité d'utiliser n'importe laquelle des transitions entre les états S et T. La probabilité d'avoir une séquence de longueur δ modélisée au sein du PDFA est la probabilité d'utiliser n'importe quelle transition et ensuite d'émettre une séquence de longueur $\delta - 1$. Nous pouvons donc écrire :

$$P(S,l = \delta) = \sum_{T \in Q} \tau_{S,T} \times P(T,l = \delta - 1). \tag{5.2}$$

C'est une suite géométrique de raison τ et de premier terme $P(S,l = 0) = \pi_F(S)$ (la probabilité que l'état S soit final). En effet, la séquence doit être acceptée par le PDFA, elle doit donc finir dans un état final. Nous obtenons donc

$$P(l = \delta) = \tau^{\delta} \times F, \tag{5.3}$$

où F est le vecteur des valeurs de $\pi_F(S)$. En utilisant l'équation 5.1 nous déduisons que :

$$\hat{E}(l) = \sum_{\delta=0}^{\infty} \delta \times \tau^{\delta} \times F. \tag{5.4}$$

Donnons un exemple afin de montrer que l'équation 5.4 permet de calculer une estimation correcte de $E(l)$. Reprenons l'ensemble de séquences déjà utilisé précédemment (table 5.1).

ab	bac	baba	abbac	abbaab
ba	abcc	bacc	abccc	baabba
abc	baab	ababc	babac	babaabc

Tab. 5.1 – *Ensemble de 15 séquences construit à partir de l'alphabet* $\Sigma = \{a,b,c\}$.

Nous déduisons de ces données que $E(l) = \frac{65}{15} = 4.33$.

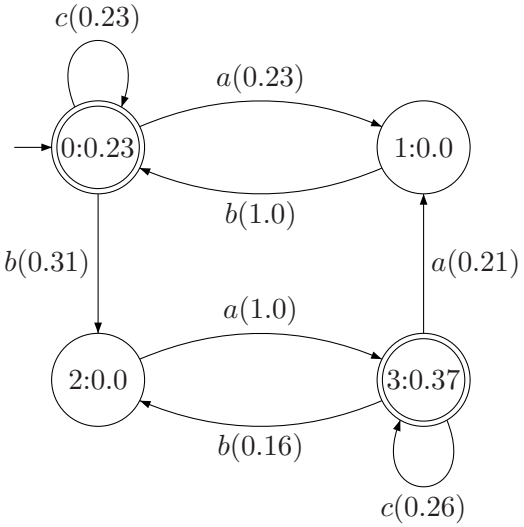

Fig. 5.2 – pdfa *inféré par* Alergia *depuis les séquences de la table 5.1.*

En appliquant la formule 5.4, à partir du pdfa de la figure 5.2, nous obtenons $\hat{E}(l) = 4.311$, qui, malgré le fait que l'on dispose d'un petit nombre de séquences ($N = 15$), est très proche de $E(l)$. Nous avons maintenant tous les outils nécessaires pour vérifier si un pdfa est assez bon pour être fouillé dans le contexte de la fouille de données séquentielles. Nous allons maintenant présenter un système complet permettant d'extraire des itinéraires fréquents dans une ville sans aucune information personnelle sur les automobilistes.

5.2 Traffic Miner

Dans le but de mettre en avant l'intérêt de notre approche dans un contexte de préservation de la vie privée, nous avons construit une application basée sur le trafic routier appelée Traffic Miner. Ce logiciel nous permet de visualiser une carte routière puis de la modéliser, à l'aide d'une interface graphique, sous la forme d'un graphe : les routes à sens unique et à double sens deviennent les transitions ; les portes d'entrées et de sorties et les croisements sont respectivement les états initiaux, finaux et simples (ni finaux, ni initiaux). L'introduction des portes d'entrées et de sorties est due au fait que la carte se limite à une fenêtre de taille fixée, il est donc nécessaire de définir des points d'entrées et de sorties sur celle-ci. Pour faciliter la lecture et l'interprétation des résultats nous avons choisi de donner un nom différent à chaque portion de rue entre deux croisements ou portes.

Fig. 5.3 – *Carte d'Arlington (USA) utilisée dans le logiciel* Traffic Miner.

La figure 5.3 décrit un exemple d'utilisation de notre logiciel sur une carte d'Ar-

lington, proche de Washington aux États-Unis. Sur chaque rue, nous disposons un compteur pour compter le nombre de voitures qui l'ont traversée (sur cette figure tout les compteurs sont à 0). Les états rouges représentent les portes de sortie et les états verts les portes d'entrée. Certains de ces compteurs sont dédiés à calculer le nombre de voitures qui ont quitté la carte (représentant les états finaux). Une fois qu'une carte est modélisée, nous pouvons simuler le trafic en générant un flot aléatoire de voitures (bouton `start`). Une distribution multinomiale est associée aux états initiaux pour choisir par quelles portes arrivent les voitures (paramétrable dans l'onglet `edit`), et à chaque croisement est aussi associée une distribution multinomiale pour choisir la route à prendre. Cette distribution est générable aléatoirement (bouton `Initialize distribution`) ou définissable par l'utilisateur pour chaque état. Une fois que la simulation a commencé, nous pouvons stopper à tout moment le flot de voitures et obtenir un PDFA $A =< Q,\Sigma,q,\pi,\pi_I,\pi_F >$ où :

- Q est l'ensemble des croisements et des portes d'entrées et de sorties,
- Σ est l'ensemble des noms de portions de rues,
- q: $Q \times \Sigma \rightarrow Q$ définit une transition, c'est-à-dire une portion de rue entre deux croisements,
- π: $Q \times \Sigma \rightarrow [0,1]$ associe une probabilité à chaque paire (S,z), c'est-à-dire la probabilité de quitter le croisement S en empruntant la route z,
- π_F: $Q \rightarrow [0,1]$ associe à chaque état final (c'est-à-dire les portes de sortie) une probabilité non nulle $\pi_F(S)$ de quitter la carte en prenant la porte S,
- π_I: $Q \rightarrow [0,1]$ associe à chaque état initial (c'est-à-dire les portes d'entrées) une probabilité non nulle $\pi_I(S)$ de rentrer dans la carte par la porte S.

D'après la définition 2.9, un PDFA doit avoir seulement un état initial pour être déterministe. Dans notre cas, malgré le fait que nous avons plusieurs états initiaux (portes d'entrée), le déterminisme n'est pas remis en cause. En effet, il n'existe pas plusieurs chemins, partant de deux états initiaux qui utilisent la même transition (rue). De plus, la somme des π_I pour tous les états initiaux est égale à 1.

La figure 5.4 montre la même carte que précédemment, où un trafic a été simulé puis stoppé lorsque 612 voitures avaient traversé la carte. On peut voir par exemple que 330 automobilistes ont emprunté la portion de route entre les états 15 et 17 (dans le coin en bas à gauche). Nous avons ensuite utilisé l'algorithme ACSM pour découvrir les motifs fréquents (bouton `Find frequent patterns`). Tous les paramètres précédemment présentés (p_0 le seuil de support, α_1 pour le test de proportion, α_2 pour le test de dépendance, θ et ϵ pour la contrainte de longueur de préfixe) sont, bien entendu paramétrables dans Traffic Miner. Dans le cas de l'expérimentation visualisée à la figure 5.4, nous avons utilisé un support de 10% et des valeurs pour les risques α_1 et α_2 également de 10%. La contrainte de longueur de préfixe n'a pas été utilisée dans cette expérimentation. Nous pouvons voir apparaître, à droite de la carte, une partie des 108 motifs fréquents et pertinents découverts par ACSM et triés par leurs fréquences d'apparition.

Traffic Miner permet également de ne conserver que les motifs maximaux, c'est-à-dire les séquences qui sont maximales dans l'ensemble des séquences fréquentes (voir

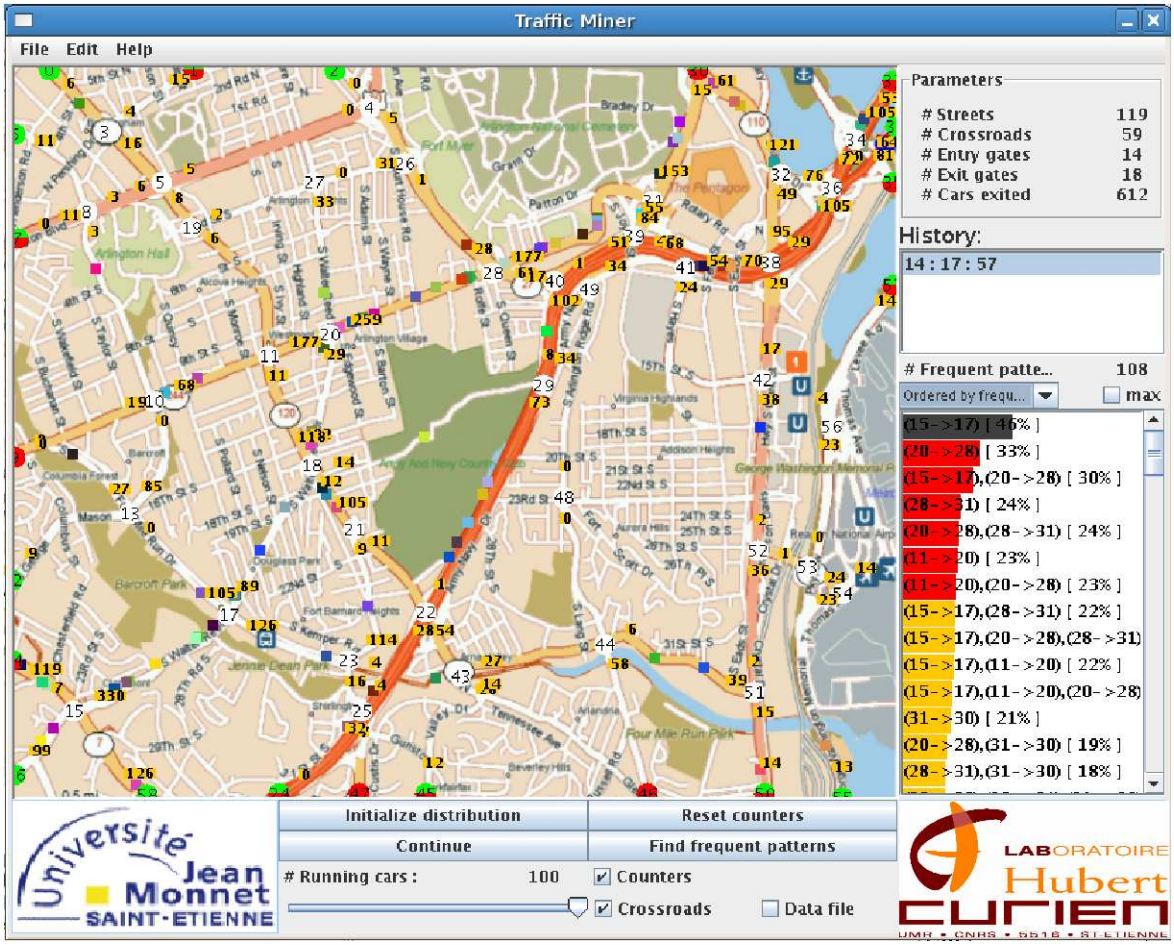

Fig. 5.4 – *Simulation du trafic sur une carte d'Arlington.*

la définition 1.7). Cet ensemble de sous-séquences est illustré à la figure 5.5, où il ne reste plus que 17 motifs maximaux, ce qui permet une meilleure lisibilité. Nous avons trié ces motifs par longueur, et la figure montre, en rouge sur la carte, le plus long motif ayant une fréquence supérieure à 10%, c'est-à-dire la plus longue succession de portions de rues (non nécessairement consécutives) empruntée par plus de 62 voitures.

5.2.1 Intérêt de la modélisation d'un flux routier

A partir du PDFA récupéré après une simulation, nous pouvons utiliser ACSM pour extraire des motifs qui peuvent être intéressants dans divers domaines. Premièrement, de tels motifs pourraient être efficacement utilisés pour réguler le trafic. En cherchant les chemins fréquemment empruntés par les automobilistes, il serait alors possible de localiser les endroits sur la carte où quelques aménagements pourraient être utiles (installation de ronds-points, de feux tricolores, etc.) pour rendre le trafic plus fluide.

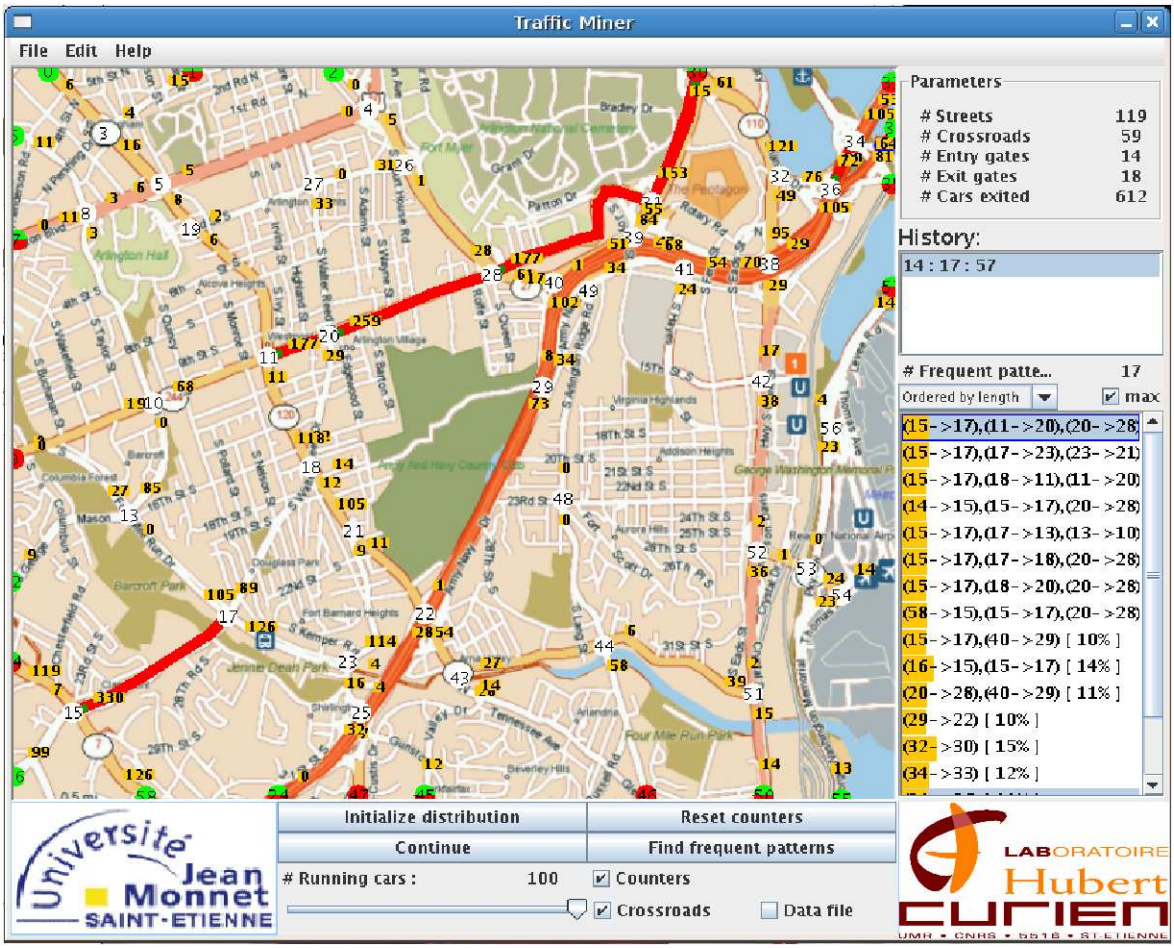

Fig. 5.5 – *Illustration des motifs maximaux.*

Par exemple, sur la figure 5.6, nous pouvons localiser, dans le coin en haut à droite de la carte, un quartier où le trafic est très dense (routes en couleur rouge). Plusieurs motifs fréquents sont localisés sur des rues voisines, ce qui permettra de définir cette zone comme prioritaire pour de futurs aménagements.

Une telle application pourrait également être utile pour simuler une nouvelle organisation du trafic (modification des sens de circulation, création de nouvelles routes, etc.) permettant ainsi de visualiser les possibles problèmes engendrés par cette nouvelle organisation (création de bouchons, de rues inutilisées, etc.). Finalement, TRAFFIC MINER pourrait servir à améliorer l'impact de campagnes publicitaires au sein d'une ville. En effet, il est important de rappeler une nouvelle fois qu'un motif fréquent et pertinent $w =< x_1 x_2 >$, extrait à l'aide de notre système, exprime le fait que la majorité des voitures qui ont emprunté la route x_1 vont probablement aussi emprunter plus tard la route x_2, non nécessairement consécutive à x_1. C'est le cas pour le motif composé de la

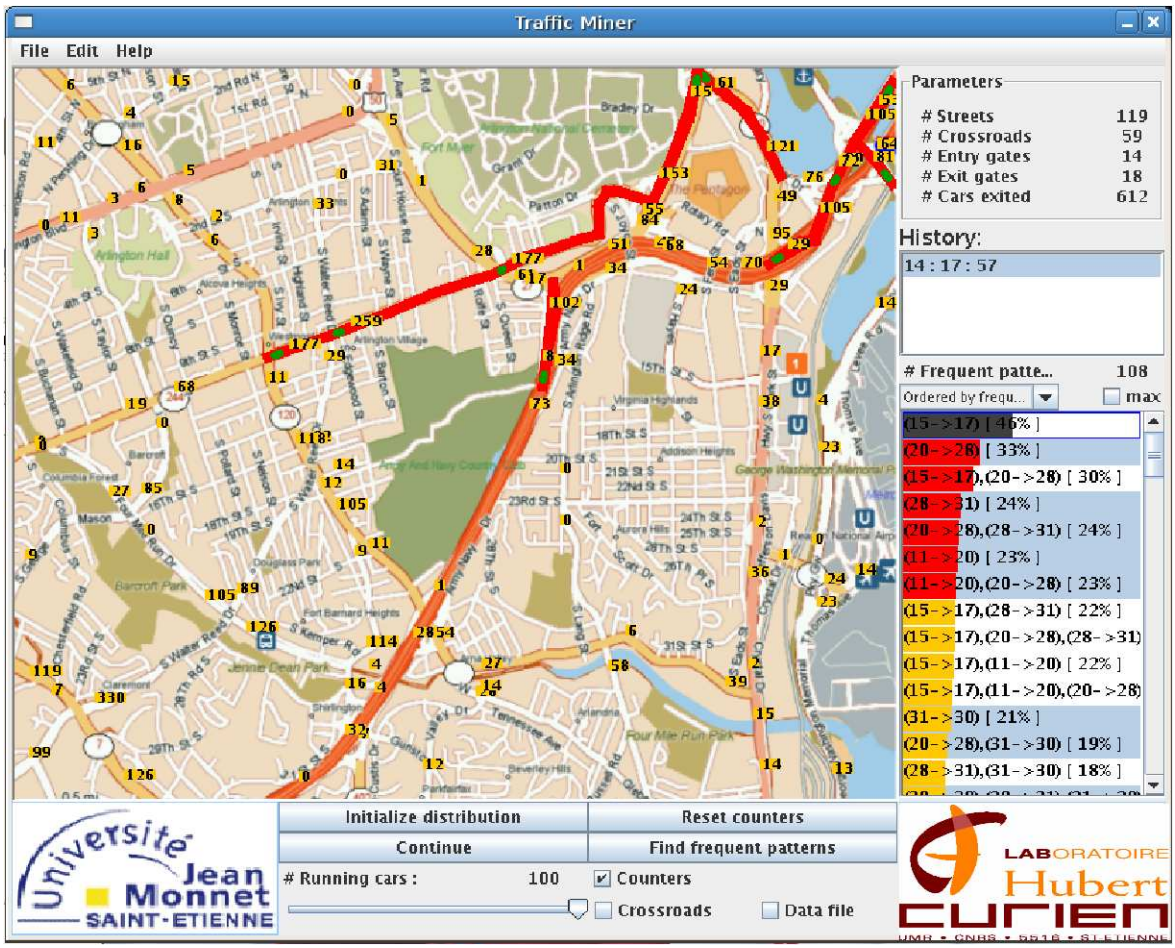

Fig. 5.6 – *Concentration de motifs fréquents.*

rue entre les états 15 et 17 et de la rue entre les états 20 et 28 dans la figure 5.7. Ce motif a un support assez conséquent de 30%. Notre système nous permet de découvrir que les individus qui ont emprunté la première rue emprunteront probablement plus tard la seconde, quels que soient les chemins qu'ils emprunteront entre ces deux segments de rues.

En pratique, plusieurs raisons peuvent expliquer cette non contiguïté des portions de rues dans le motif. Certaines personnes peuvent, par exemple, prendre la rue entre les états 17, 18, 11 et 20 pour déposer leurs enfants à l'école. D'autres peuvent préférer emprunter les rues entre les états 17, 18 et 20 car il constitue l'itinéraire le plus court. Finalement, une dernière catégorie d'automobilistes peut choisir les rues entre les états 17, 13, 10, 11 et 20 pour éviter les embouteillages bien que ce chemin soit le plus long. Mais, au final, tous les automobilistes se retrouvent sur la portion de rue entre les états 20 et 28. Ce type d'information pourrait, entre autres, aider un publi-

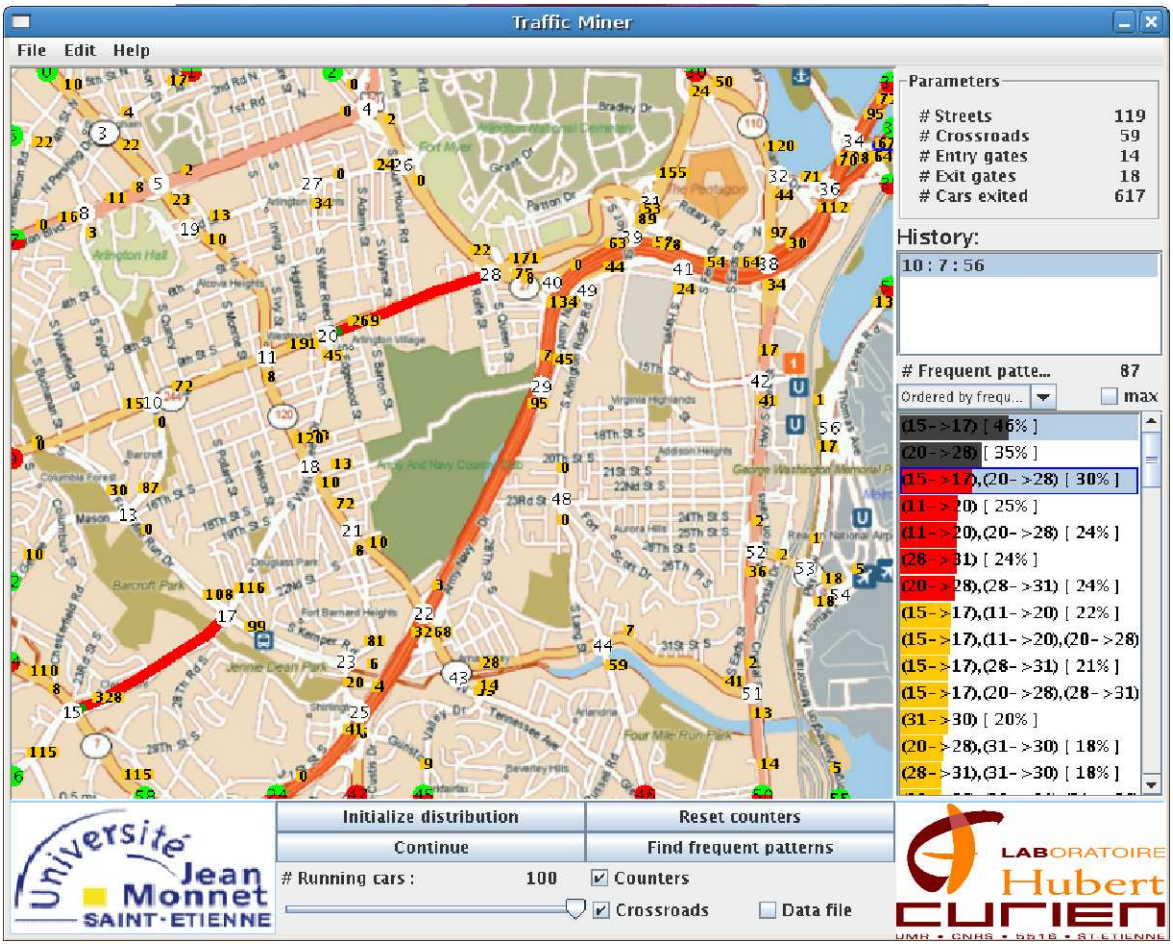

Fig. 5.7 – *Illustration des motifs non consécutifs.*

ciste à trouver les endroits stratégiques pour positionner les panneaux publicitaires. Par exemple, à l'heure actuelle, les campagnes publicitaires sont parfois constituées d'affiches différentes avec un fil conducteur entre elles. Donc, connaître les différentes routes fréquemment empruntées par les mêmes automobiliste peut permettre au publiciste de choisir les positions stratégiques pour que ceux-ci puissent voir les différents panneaux. De même, dans le cas d'une campagne comportant une unique affiche, il peut utiliser ces informations pour choisir les emplacements qui accroîtraient l'impact de la campagne. Dans le cas de la figure 5.7, on imagine qu'il pourrait par exemple vouloir placer une affiche sur la portion de rue entre les états 15 et 17, puis une autre dans la portion de rue entre les états 20 et 28.

La contrainte de longueur de préfixe que nous avons mise en place peut aussi trouver son utilité dans le cadre du trafic routier. Dans la figure 5.8, nous montrons le résultat obtenu pour la recherche de sous-séquences fréquentes et pertinentes ayant un support

FIG. 5.8 – *Illustration de la contrainte de longueur de préfixe.*

supérieur à 5% avec une longueur de préfixe comprise entre 1 et 5 (soit $\theta = 3$ et $\epsilon = 2$). Cette contrainte peut, par exemple, être utile pour vérifier si les axes principaux sont bien desservis à partir des portes d'entrée de la ville. C'est ce qui est illustré dans cette figure. Les portes d'entrée (en vert) ont été placées sur la partie basse de la carte, et l'on peut constater que les automobilistes se retrouvent, après un maximum de 5 tronçons de rues sur la voie rapide (routes rouges entre les intersections 20, 27 et 36), ou à proximité.

5.3 Améliorations calculatoires

Dans cette section, nous montrons que la technique, présentée au chapitre 3, permettant d'estimer la probabilité d'un motif à partir d'un PDFA peut être améliorée algorithmiquement. Lors du développement de TRAFFIC MINER, nous avons donc tenté

d'améliorer la complexité calculatoire de ces estimations. En effet, nous avons pu voir qu'elles requièrent l'utilisation de la méthode de Cramer, ce qui impose d'inverser des matrices pour résoudre un système d'équations linéaires. Par exemple, pour calculer la probabilité d'un motif de taille n, constitué de m lettres différentes, il faut calculer une inversion de matrice pour chacun des m symboles du motif. Cette méthode nécessite, de plus, que les matrices soient inversibles et nous avons pu constater dans nos premières expérimentations que ce n'était pas toujours le cas. Pour palier à ce problème et améliorer la complexité calculatoire des estimations, nous proposons d'utiliser la méthode de la factorisation LU qui évite les inversions de matrices. Cette technique d'algèbre linéaire consiste à décomposer une matrice comme le produit de deux matrices : L une matrice triangulaire inférieure avec des 1 sur la diagonale et U une matrice triangulaire supérieure[1].

Rappelons que pour estimer la probabilité d'un motif contenant une lettre x, l'expression suivante est utilisée :

$$P(S,x) = \pi(S,x) + \sum_{T \in Q} \left(\sum_{z \neq x, q(S,z)=T} \pi(S,z) \right) \times P(T,x). \tag{5.5}$$

On introduit ensuite la matrice ρ telle que

$$\rho_{S,T}(x) = \sum_{z \neq x, q(S,z)=T} \pi(S,z). \tag{5.6}$$

Le but est de résoudre cette équation et de trouver les valeurs de $P(S,x)$ pour chaque état $S \in Q$. Les équations 5.5 et 5.6 aboutissent à

$$P(x) = \pi(x) + \rho(x) \times P(x),$$

donc

$$P(x) - \rho(x) \times P(x) = \pi(x),$$

et enfin

$$(I - \rho(x)) \times P(x) = \pi(x).$$

Nous avons donc besoin de résoudre l'équation $AX = B$ où A est la matrice $(I - \rho)$, B est le vecteur π et X est le vecteur des valeurs inconnues $P(x)$. La factorisation LU consiste alors à construire les matrices L et U à partir de la matrice A généralement en utilisant la méthode du pivot de Gauss. On résout ensuite le système d'équations en deux étapes, tout d'abord en résolvant $Lz = B$, puis $UX = z$. Reprenons le calcul de $P(c)$ présenté précédemment à la section 3.2.2. Il faut trianguliariser la matrice A pour trouver L et U, grâce à la méthode des pivots de Gauss :

[1] Noter que pour nos expérimentations, nous avons utilisé une librairie de résolution de matrice appelée Meschach [SL94].

$$A = I - \rho(c) = \begin{pmatrix} 1 & -0.23 & -0.31 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & -0.21 & -0.16 & 1 \end{pmatrix}$$

Nous allons, étape par étape, annuler les valeurs inférieures gauches de la matrice A en faisant des combinaisons linéaires des lignes de la matrice, pour obtenir la matrice triangulaire supérieur U. La matrice L est construite avec les facteurs utilisés dans les combinaisons linéaires. Notons qu'il faut toujours raisonner en terme de soustraction pour obtenir les facteurs de la matrice L. Il faut, premièrement, annuler la valeur $A_{2,1} = -1$. Nous allons donc effectuer l'opération $A_2 + A_1$, soit $A_2 - (-1 \times A_1)$, donc $L_{2,1} = -1$. On obtient :

$$U = \begin{pmatrix} 1 & -0.23 & -0.31 & 0 \\ 0 & 0.77 & -0.31 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & -0.21 & -0.16 & 1 \end{pmatrix}.$$

On annule ensuite $U_{4,2}$ en effectuant $U_4 + 0.27 \times U_2$, d'où $L_{4,2} = -0.27$:

$$U = \begin{pmatrix} 1 & -0.23 & -0.31 & 0 \\ 0 & 0.77 & -0.31 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & -0.244 & 1 \end{pmatrix}.$$

Il faut enfin annuler $U_{4,3}$ en effectuant $U_4 + 0.244 \times U_3$, d'où $L_{4,3} = -0.244$:

$$U = \begin{pmatrix} 1 & -0.23 & -0.31 & 0 \\ 0 & 0.77 & -0.31 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0.755 \end{pmatrix}.$$

La matrice L contient donc des 1 sur la diagonale et les valeurs définies précédemment lors de la construction de U, soit

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -0.27 & -0.244 & 1 \end{pmatrix}.$$

Il faut ensuite résoudre $Lz = B$. De par la forme de L, cette résolution est immédiate :

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -0.27 & -0.244 & 1 \end{pmatrix} \times \begin{pmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{pmatrix} = \begin{pmatrix} 0.23 \\ 0 \\ 0 \\ 0.26 \end{pmatrix},$$

d'où

$$z = \begin{pmatrix} 0.23 \\ 0.23 \\ 0 \\ 0.322 \end{pmatrix}.$$

Nous résolvons ensuite $UX = z$. Là aussi, de par la forme de U, la résolution est triviale :

$$U = \begin{pmatrix} 1 & -0.23 & -0.31 & 0 \\ 0 & 0.77 & -0.31 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0.755 \end{pmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0.23 \\ 0.23 \\ 0 \\ 0.322 \end{pmatrix},$$

d'où :

$$P(c) = X = \begin{pmatrix} 0.47 \\ 0.47 \\ 0.426 \\ 0.426 \end{pmatrix}.$$

Nous pouvons vérifier que les résultats sont identiques à ceux présentés à la section 3.2.2.

Pour adapter cette méthode à notre contexte, nous avons donc réécrit les équations d'Hingston. A l'heure actuelle, nous n'avons pas trouvé de méthode permettant de résoudre plus efficacement le calcul de la matrice F (voir l'équation 3.3). En effet, F est une matrice et non un vecteur comme P ; la méthode de la factorisation LU n'est donc pas applicable. Pour contourner ce problème, nous avons réécrit les formules permettant de calculer la probabilité d'un motif $< xyz >$, donc de n'importe quel motif :

$$\begin{aligned} P(S, < xyz >) &= \sum_{z \neq x \in \Sigma} (\pi(S,z) \times P(q(S,z), < xyz >)) \\ &+ \pi(S,x) \times P(q(S,x), < yz >). \end{aligned} \tag{5.7}$$

En utilisant la forme matricielle, nous obtenons :

$$P(< xyz >) = \rho(x) \times P(< xyz >) + \pi(x) \times P(< yz >),$$

donc

$$(I - \rho(x)) \times P(< xyz >) = \pi(x) \times P(< yz >).$$

Nous pouvons donc résoudre l'équation grâce à la factorisation LU pour trouver $P(< xyz >)$ car, cette fois, le terme $\pi(x) \times P(< yz >)$ est un vecteur. Notons que du fait de l'anti-monotonie de la contrainte de fréquence, lors du calcul de la probabilité du motif $< xyz >$, la probabilité du sous-motif $< yz >$ sera connue.

Nous avons également introduit, au chapitre 4.4.2, de nouvelles formules permettant de prendre en compte des contraintes de longueur de préfixe. Comme pour les formules

précédentes, la formule 4.27 nécessite des inversions de matrice. Or, nous pouvons réécrire cette formule comme suit :

$$P(< x_1...x_l >,\theta) = \tau^{\theta} \times \gamma(x_1) \times P(< x_2...x_l >). \tag{5.8}$$

Nous pouvons ainsi utiliser la méthode présentée pour l'équation 5.7 afin de calculer la probabilité de l'équation 5.8.

Cette méthode de calcul peut efficacement être utilisée si, pour le calcul de la probabilité d'un motif $w =< w_1 \dots w_n >$, la probabilité du préfixe $< w_1 \dots w_{n-1} >$ a déjà été calculée. Or, comme l'algorithme ACSM fonctionne avec une recherche par niveau, cette condition est vérifiée (voir l'algorithme 4.2 de génération des candidats).

De plus, nous avons utilisé, pour le codage des programmes, des matrices creuses, c'est-à-dire des matrices où seules les valeurs non nulles sont codées[2]. En effet, les matrices sont construites à partir des transitions de l'automate qui ne sont généralement pas extrêmement nombreuses pour un état donné. Sur une ligne d'une matrice, représentant toutes les transitions d'un état à tous les autres, le nombre de 0 est généralement assez important. Nous pouvons d'ailleurs le remarquer sur l'ensemble des exemples que nous avons présentés.

5.4 Comparaisons entre TRAFFIC MINER et SPAM

Nous réalisons, dans cette section, une étude expérimentale visant à comparer notre approche basée sur les PDFAs à l'algorithme SPAM. Le but est de montrer que notre méthode peut être plus efficace en temps de calcul lorsque l'on dispose d'une représentation graphique des données, ce qui est le cas avec TRAFFIC MINER.

Comme nous ne disposons pas des données d'origine, une comparaison avec des algorithmes classique de fouille de données séquentielles paraît difficile. Pour contourner ce problème, nous avons généré un ensemble de séquences à partir de l'automate issu de la carte de la figure 5.8 à un instant t. Nous pouvons alors comparer les résultats obtenus avec SPAM sur ces séquences et ceux obtenus par ACSM directement sur l'automate. Le protocole expérimental est donc le suivant : nous simulons un flux de voitures sur notre carte, et le stoppons pour obtenir un PDFA. Nous utilisons ensuite ACSM pour extraire les motifs fréquents (ici, nous n'utilisons pas les contraintes de pertinence et de longueur de préfixe pour permettre une comparaison avec SPAM). Durant cette expérience, nous mesurons la complexité en temps (appelée ACSM time sur la figure 5.9). Notons que nous n'avons pas à prendre en compte le temps concernant la construction de l'automate puisqu'il n'est pas appris mais est donné par l'application. Directement utilisé sur un PDFA, ACSM ne dépend pas du nombre de séquences, ce qui explique le temps constant sur la figure 5.9.

À partir du PDFA, nous échantillonnons plusieurs ensembles de séquences (contenant de 100 à 35 000 séquences). Pour chaque échantillon, nous utilisons SPAM pour extraire les motifs fréquents. Pour ce faire, nous utilisons un support de $p_0 = 10\%$.

[2] Cette fonction est aussi implémentée dans la librairie Meschach.

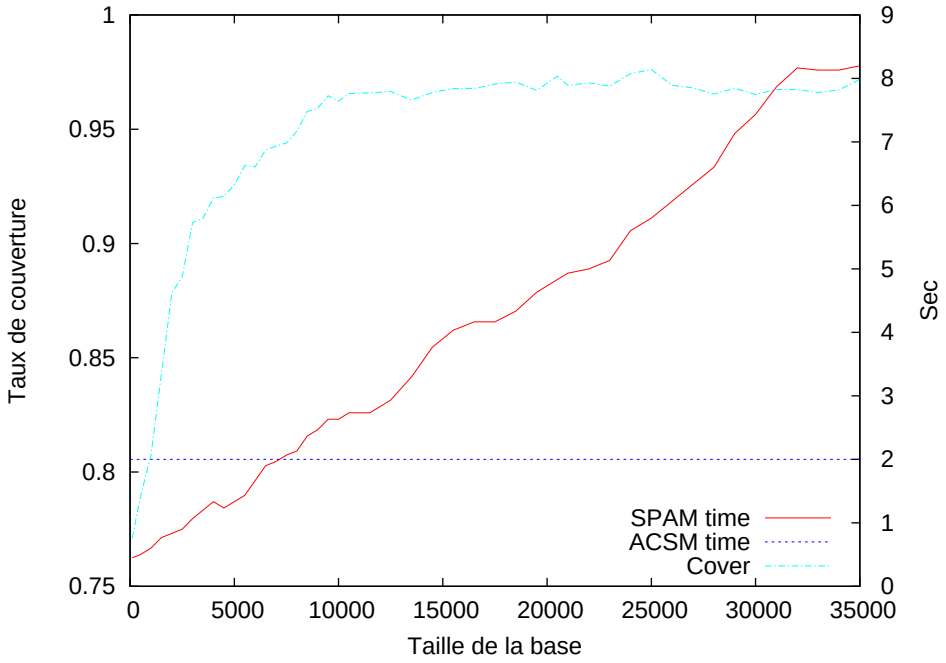

Fig. 5.9 – *Comparaison entre* spam *et* acsm

Nous mesurons également la complexité en temps de spam en prenant en compte le temps d'échantillonnage et le temps de la fouille (notée spam time sur la figure 5.9). Les courbes de la figure 5.9 décrivent le comportement des deux méthodes. Tandis que acsm a une complexité en temps constante, celle de spam augmente naturellement avec l'accroissement du nombre de séquences à traiter. Nous avons ajouté une courbe (Cover) sur la figure, qui correspond à la proportion de motifs fréquents extraits par spam qui ont aussi été extraits par acsm. L'objectif de cette courbe est de pouvoir déterminer le nombre minimum de séquences nécessaires pour obtenir avec spam les mêmes motifs qu'avec acsm. Nous pouvons observer que spam a besoin d'un grand nombre de séquences pour approcher les résultats de acsm. Nous pouvons aussi noter que lorsque la taille de l'ensemble est suffisamment grande, environ 15 000 exemples, la plupart des motifs extraits à partir du pdfa sont couverts par les motifs fréquents trouvés par spam (environ 96%). Mais, dans ce cas, le coût de spam d'un point de vue du temps de calcul est plus important que celui de acsm.

5.5 Conclusion

Dans ce chapitre, nous avons pu montrer que la fouille de données séquentielles à partir d'automates constitue une solution élégante au problème du respect de la vie privée. Nous avons vu que, dans le cas des problèmes pouvant se modéliser sous la forme de flux d'informations dans un graphe, acsm permet d'extraire des motifs séquentiels

fréquents sans avoir accès aux séquences d'origine, c'est-à-dire sans exploiter d'information à caractère privé des utilisateurs, concernés par les séquences sous-jacentes. Dans le cas des flux routiers, il n'est alors plus nécessaire de suivre les automobilistes individuellement avec des techniques intrusives. Dans le cas des usages du Web, nous n'avons pas besoin de mémoriser les numéros IP des internautes visitant les sites, ni de pré-traiter les fichiers logs souvent de très mauvaise qualité.

Les séquences d'origine n'étant plus disponibles, nous avons proposé un nouveau critère de qualité (la taille moyenne) associé à l'automate qui permet de juger de la capacité de celui-ci à être un bon estimateur des probabilités des motifs.

Le contexte de l'étude de flux routiers nous a également permis de mettre en avant l'intérêt des contraintes de pertinence et de longueur de préfixe. Dans le cadre d'autres applications, il est fort probable que de nouveaux types de contraintes seront utiles et il sera alors nécessaire de les mettre en place dans ACSM. Par exemple, si nous reprenons le cas présenté précédemment de l'étude du comportement des utilisateurs d'un site Web, il pourrait être judicieux de trouver tous les motifs (c'est-à-dire les parcours de navigation) qui ne contiennent pas une page donnée (par exemple la page de vente). Ainsi, l'administrateur pourra trouver les raisons qui font que les internautes n'aboutissent pas à cette page. La non appartenance d'un item à un motif est donc un exemple, parmi d'autres, de contrainte qu'il sera intéressant d'intégrer.

Nous avons enfin montré, dans ce chapitre, comment mettre en œuvre des techniques calculatoires efficaces permettant à ACSM d'être plus efficace que SPAM en temps de traitement, notamment dans le cas où l'automate n'a pas à être inféré mais est donné directement par l'application (graphe d'un site Web, graphe d'un réseau routier, etc.).

Conclusion générale et perspectives

Dans le cadre de cette thèse, nous avons tenté l'exercice délicat consistant à établir des liens entre les modèles obtenus par des algorithmes d'inférence grammaticale et la connaissance induite par des techniques de fouille de données séquentielles. Partant du constat que le point commun entre ces deux contextes différents de travail est la manipulation de données structurées sous forme de séquences de symboles, nous avons tenté d'exploiter les propriétés des PDFAs inférés à partir de ces séquences au profit d'une fouille de données séquentielles plus efficace.

Dans ce contexte, nous avons montré que l'exploitation brute, non seulement des séquences d'origine mais aussi des automates probabilistes inférés à partir de celles-ci, ne garantit pas forcément une extraction de connaissance pertinente. Nous avons apporté dans cette thèse plusieurs contributions, sous la forme de bornes minimales et de contraintes statistiques, permettant ainsi d'assurer une exploitation fructueuse des séquences et des PDFAs.

Dans un premier temps, nous avons pris en compte le fait que les données à notre disposition sont avant tout **issues d'une distribution statistique cible inconnue**. Nous avons tout d'abord produit une condition sur le nombre de séquences nécessaires pour contrôler, à l'issue d'un processus de fouille, les taux de faux négatifs et de faux positifs. A notre connaissance, ce résultat théorique, basé sur les risques de type I et II d'un test de proportion, est une première dans le domaine de la fouille de données séquentielles, et il ouvre la voie à plusieurs extensions prometteuses. La première porte sur l'exploitation de connaissances sur **la distribution empirique** des séquences à notre disposition. En effet, nous avons montré qu'il était nécessaire, pour calculer la borne, de choisir des valeurs de rejet du test de support. Une perspective possible est de pondérer ces valeurs de rejet par la quantité de motifs dans les données qui ont pour support cette valeur de rejet. Nous avons déjà évoqué les difficultés que pose le

passage au cas continu, et nous devrons également aborder la fiabilité des estimations issues de distributions empiriques. Ici aussi, le nombre de séquences sera un paramètre important. D'autre part, nous nous sommes restreints, dans cette thèse, à l'étude des sources d'erreurs dans le cadre de la fouille de données séquentielles. Or, les mêmes problèmes se posent pour la fouille de données transactionnelles et **l'extraction de règles d'association**, où la distribution sous-jacente doit aussi être considérée pour extraire les motifs pertinents. Nous envisageons donc d'adapter la méthode ainsi présentée à la recherche de telles règles. Pour cela, il sera nécessaire de concevoir une approche permettant de modéliser les données comportant des itemsets de taille supérieure à 1, ce qui n'est actuellement pas le cas pour les PDFAs tels que nous les avons présentés. De plus, dans le cadre des règles d'association, le critère de sélection n'est plus uniquement le seuil de support. En effet, les règles, pour être extraites, doivent aussi respecter un seuil de confiance. Il sera donc nécessaire de trouver les tests statistiques appropriés à ce nouveau critère.

Si la borne que nous avons proposée n'est pas respectée par les contraintes de l'application considérée, nous avons alors étudié les possibilités d'utiliser les PDFAs pour généraliser les données et ainsi produire des résultats plus proches de la distribution cible. Ces automates ont aussi l'avantage de constituer une représentation condensée des données. Ces deux constats nous ont permis de considérer que les automates probabilistes constituaient des bons candidats pour modéliser des séquences destinées à être fouillées. Dans le contexte de l'apprentissage d'automates probabilistes à des fins de fouille, nous avons fourni **une borne minimale sur le nombre de symboles** nécessaires pour assurer une bonne fusion, effectuée au cours de l'algorithme ALERGIA. Notre objectif est de continuer à travailler sur l'amélioration des règles de fusion d'états. Nous avons pu voir qu'ALERGIA prenait des décisions très locales (le test de Hoeffding est réalisé sur chaque transition). Des améliorations ont déjà été proposées dans la littérature, soit en exploitant globalement toutes les transitions sortantes des états candidats (voir [KD02]), soit en calculant en même temps une information plus globale, comme cela est réalisé dans MDI [TDdlH00] avec le calcul de perplexité sur un échantillon test. Afin d'étendre la relation bilatérale entre les algorithmes d'inférence grammaticale et les approches de fouille de données séquentielles, nous envisageons **d'utiliser les motifs les plus fréquents (extraits par exemple par SPAM) pour valider les différentes fusions** réalisées par l'algorithme ALERGIA. Le principe serait basé sur le fait que si la fusion de deux états entraîne de trop grandes modifications dans l'estimation des probabilités des motifs fréquents, alors celle-ci ne sera pas acceptée.

Nous avons présenté dans cette thèse les outils mathématiques nécessaires au calcul, à partir d'un PDFA, des estimations des probabilités d'apparition de n'importe quel type de motif constitué de lettres non-consécutives. Ces outils avaient été initialement présentés par HINGSTON [Hin02], et nous les avons étendus à une plus grande diversité de motifs, notamment dans le cadre d'extractions sous contraintes. Dans ce contexte, nous avons introduit deux nouvelles contraintes. La première permet

d'extraire des motifs statistiquement pertinents et fait appel une nouvelle fois à la statistique inférentielle. La seconde agit sur la structure des motifs, plus précisément sur une longueur de préfixe avant le motif recherché. Nous étudierons, dans la suite de ce travail, la possibilité **d'introduire de nouvelles contraintes**, en phase avec celles déjà exploitées dans le cadre classique de la fouille de données séquentielles, comme c'est le cas des contraintes de longueur, d'appartenance, d'agrégats, de similarité, etc.

Enfin, nous avons montré que dans le cadre de l'étude des flux de données, notre approche fournissait un outil tout à fait intéressant pour préserver la vie privée des individus. En effet, nous avons pu voir que le problème étudié peut souvent être directement représenté, grâce à des simples procédures de comptage, par des automates probabilistes, sans faire appel aux séquences d'origine qui divulgueraient des informations privées des individus concernés. Nous avons exploité cette idée dans le contexte particulier des trafics routiers urbains, même si celle-ci peut s'adapter également aux problématiques du Web, par exemple. Nous avons montré dans ce cadre l'intérêt des différentes contraintes et de l'utilisation des PDFAs pour la préservation de la vie privée des automobilistes. L'étape suivante va consister à tenter de valoriser ce travail en présentant notre prototype à de futurs partenaires potentiels (DDE, publicitaires, entreprises de développement de sites Web).

Bibliographie

[ABE+99] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios. Disclosure limitation of sensitive rules. In *Proceedings of Knowledge and Data Engineering Exchange Workshop (KDEX'99)*, pages 45–52, Chicago, IL, USA, November 1999.

[AFGY02] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu. Sequential pattern mining using a bitmap representation. In *Proceedings of the 8th ACM SIGMOD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pages 429–435, Edmonton, Alberta, Canada, July 2002. ACM Press.

[ALB03a] H. Albert-Lorincz and J-F. Boulicaut. A framework for frequent sequence mining under generalized regular expression constraints. In *Proceedings of the 2nd International Workshop on Knowledge Discovery in Inductive Databases (KDID'03)*, pages 2–16, Cavtat-Dubrovnik, Croatia, September 2003. co-located with ECML-PKDD 2003.

[ALB03b] H. Albert-Lorincz and J-F. Boulicaut. Mining frequent sequential patterns under regular expressions: a highly adaptative strategy for pushing constraints. In *Proceedings of the 3rd SIAM International Conference on Data Mining (SDM'03)*, pages 316–320, San Francisco, USA, May 2003.

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In J. B. Bocca, M. Jarke, and C. Zaniolo, editors, *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94)*, pages 487–499, Santiago de Chile, Chile, September 1994. Morgan Kaufmann.

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. In P. S. Yu and A. L. P. Chen, editors, *Proceedings of the 11th International Conference on Data Engineering (ICDE'95)*, pages 3–14, Taipei, Taiwan, March 1995. IEEE Computer Society.

[AS00] R. Agrawal and R. Srikant. Privacy-preserving data mining. In W. Chen, J. F. Naughton, and P. A. Bernstein, editors, *Proceedings of the 19th ACM SIGMOD Conference on Management of Data*, pages 439–450, Dallas, Texas, USA, May 2000. ACM.

[AU72] A. V. Aho and J. D. Ullman. *The theory of parsing, translation, and compiling.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1972.

[AY08] C. C. Aggarwal and P. S. Yu. *Privacy-Preserving Data Mining: Models and Algorithms*. Springer Publishing Company, 2008.

[BGKW02] C. Bucila, J. Gehrke, D. Kifer, and W. M. White. Dualminer: a dual-pruning algorithm for itemsets with constraints. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'02)*, pages 42–51, Edmonton, Alberta, Canada, July 2002. ACM.

[BJ05] J-F. Boulicaut and B. Jeudy. Constraint-based data mining. In O. Maimon and L. Rokach, editors, *The Data Mining and Knowledge Discovery Handbook*, pages 399–416. Springer, 2005.

[BL98] J. Borges and M. Levene. Mining association rules in hypertext databases. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 149–153, New York City, New York, USA, August 1998. AAAI Press.

[BL99] J. Borges and M. Levene. Data mining of user navigation patterns. In B. M. Masand and M. Spiliopoulou, editors, *Revised Papers from the International Workshop on Web Usage Analysis and User Profiling (WEBKDD '99)*, volume 1836 of *Lecture Notes In Computer Science*, pages 92–111, San Diego, California, USA, 1999. Springer.

[BL04] J. Borges and M. Levene. A dynamic clustering-based markov model for web usage mining. *CoRR: The Computing Research Repository*, cs.IR/0406032, June 2004.

[BL05] J. Borges and M. Levene. Generating dynamic higher-order markov models in web usage mining. In A. Jorge, L. Torgo, P. Brazdil, R. Camacho, and J. Gama, editors, *Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'05)*, volume 3721 of *Lecture Notes in Computer Science*, pages 34–45, Porto, Portugal, October 2005. Springer.

[BL07] J. Borges and M. Levene. Evaluating variable-length markov chain models for analysis of user web navigation sessions. *IEEE Transactions on Knowledge and Data Engineering*, 19(4):441–452, April 2007.

[BMS08] F. Bonchi, B. Malin, and Y. Saygin. Recent advances in preserving privacy when mining data. *Data Knowledge Engineering*, 65(1):1–4, 2008.

[Cal07] J. Callut. *First passage times dynamics in Markov Models with applications to HMM : induction, sequence classification and graph mining*. PhD thesis, Catholic University of Leuven, 2007.

[CFCK98] C.H. Cai, A.W.C. Fu, C.H. Cheng, and W.W. Kwong. Mining association rules with weighted items. In B. Eaglestone, B. C. Desai, and J. Shao, editors, *IDEAS '98: Proceedings of the 1998 International Symposium on Database Engineering & Applications*, pages 68–77, Cardiff, Wales, U.K., July 1998. IEEE Computer Society.

[Cho57] N. Chomsky. *Syntactic structures*. Mouton, 1957.

[CKV+03] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Zhu. Tools for privacy preserving distributed data mining. *ACM SIGKDD Explorations*, 4(2):28–34, January 2003.

[CM00] L. Chang and I. S. Moskowitz. An integrated framework for database privacy protection. In B. M. Thuraisingham, R. P. Riet, K. R. Dittrich, and Z. Tari, editors, *Proceedings of the IFIP TC11/ WG11.3 14th Annual Working Conference on Database Security*, volume 201 of *IFIP Conference Proceedings*, pages 161–172, Deventer, The Netherlands, The Netherlands, August 2000. Kluwer, B.V.

[CM02] A. Cornuéjols and L. Miclet. *Apprentissage artificiel : Concepts et algorithmes*. Eyrolles, 2002.

[CMB02] M. Capelle, C. Masson, and J-F. Boulicaut. Mining frequent sequential patterns under a similarity constraint. In *Proceedings of the 3rd International Conference on Intelligent Data Engineering and Automated Learning (IDEAL '02)*, volume 2412 of *Lecture Notes In Computer Science*, pages 1–6, Manchester, UK, August 2002. Springer.

[CO94] R. C. Carrasco and J. Oncina. Learning stochastic regular grammars by means of a state merging method. In R. C. Carrasco and J. Oncina, editors, *Proceedings of the 2nd International Colloquium on Grammatical Inference (ICGI'94)*, volume 862 of *Lecture Notes In Computer Science*, pages 139–152, Alicante, Spain, September 1994. Springer-Verlag.

[CT04] A. Clark and F. Thollard. Pac-learnability of probabilistic deterministic finite state automata. *The Journal of Machine Learning Research*, 5:473–497, December 2004.

[CWC04] D. Chiu, Y. Wu, and A. L. P. Chen. An efficient algorithm for mining frequent sequences by a new strategy without support counting. In *Proceedings of the 20th International Conference on Data Engineering (ICDE '04)*, page 375, Houston, Texas, USA, February 2004. IEEE Computer Society.

[DA01] W. Du and M. J. Atallah. Secure multi-party computation problems and their applications: A review and open problems. In *Proceedings of the 2001 workshop on New security paradigms (NSPW '01)*, pages 13–22, Cloudcroft, New Mexico, USA, September 2001. ACM.

[DCD+06] P. Dupont, J. Callut, G. Dooms, J-N. Monette, and Y. Deville. Relevant subgraph extraction from random walks in a graph. Technical Report 2006-2007, UCL/FSA/INGI, November 2006.

[dlH05] C. de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 38(9):1332–1348, 2005.

[DM98] P. Dupont and L. Miclet. Inférence grammaticale régulière : fondements théoriques et principaux algorithmes. Technical report, INRIA, RR-2449, July 1998.

[Fis22] R. A. Fisher. On the interpretation of chi-square from the contingency tables, and the calculation of P. *Journal of the Royal Statistical Society.*, 85:87–94, 1922.

[FPSM91] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus. Knowledge discovery in databases: An overview. In G. Piatetsky-Shapiro and W. J. Frawley, editors, *Knowledge Discovery in Databases*, pages 1–30. AAAI/MIT Press, 1991.

[FPSS96] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In U. M. Fayyad, G. Piatetsky-Shapiro, P.c Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI/MIT Press, Menlo Park, CA, USA, 1996.

[GH06] L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Survey*, 38(3), September 2006.

[GHP07] G. Gidófalvi, X. Huang, and T. Bach Pedersen. Privacy-preserving data mining on moving object trajectories. In C. Becker, C. S. Jensen, J. Su, and D. Nicklas, editors, *Proceedings of the 8th International Conference on Mobile Data Management (MDM'07)*, pages 60–68, Mannheim, Germany, May 2007. IEEE.

[GMMT06] A. Gionis, H. Mannila, T. Mielikäinen, and P. Tsaparas. Assessing data mining results via swap randomization. In T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, editors, *Proceedings of the 12th International Conference on Knowledge Discovery and Data Mining (KDD'06)*, pages 167–176, Philadelphia, PA, USA, August 2006. ACM.

[Gol67] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[Gol78] E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37(3):302–320, 1978.

[GRS02] M. Garofalakis, R. Rastogi, and K. Shim. Mining sequential patterns with regular expression constraints. *IEEE Transactions on Knowledge and Data Engineering*, 14(3):530–552, May/June 2002.

[Hin02] P. Hingston. Using finite state automata for sequence mining. In *Proceedings of the 25th Australasian Conference on Computer Science (ACSC'02)*, pages 105–110, Melbourne, Australia, 2002. Australian Computer Society, Inc.

[Hoe63] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.

[HPMA+00] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.-C. Hsu. Freespan: Frequent pattern-projected sequential pattern mining. In *Proceedings of the 6th ACM SIGMOD International Conference on Knowledge Discovery and Data Mining (KDD' 00)*, pages 355–359, Boston, MA, USA, August 2000. ACM Press.

[HT00] C. De La Higuera and F. Thollard. Identification in the limit with probability one of stochastic deterministic finite automata. In A. L. Oliveira, editor, *Proceedings of the 5th International Colloquium on Grammatical Inference (ICGI '00)*, volume 1891 of *Lecture Notes In Computer Science*, pages 141–156, London, UK, September 2000. Springer.

[JM80] F. Jelinek and R. L. Mercer. Interpolated estimation of markov source parameters from sparse data. In E. Gelsema and L. Kanal, editors, *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397, Amsterdam, The Netherlands: North-Holland, May 1980.

[Kat87] S. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech, and Signal Processing (ASSP)*, 35(3):400–401, March 1987.

[KB00] R. Kosala and H. Blockeel. Web mining research: A survey. *SIGKDD Explorations Newsletter*, 2(1):1–15, June 2000.

[KC04] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1026–1037, September 2004.

[KD02] C. Kermorvant and P. Dupont. Stochastic grammatical inference with multinomial tests. In P. W. Adriaans, H. Fernau, and M. van Zaanen, editors, *Proceedings of the 6th International Colloquium on Grammatical Inference (ICGI'02)*, volume 2484 of *Lecture Notes in Computer Science*, pages 149–160, Amsterdam, The Netherlands, September 2002. Springer.

[KL51] S. Kullback and R. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, March 1951.

[KMR+94] M. Kearns, Y. Mansour, D. Ron, R. Rubinfeld, R. E. Schapire, and L. Sellie. On the learnability of discrete distributions. In *Proceedings of the 26th annual ACM symposium on Theory of computing (STOC '94)*, pages 273–282, Montreal, Quebec, Canada, May 1994. ACM.

[KMT99] M. Klemettinen, H. Mannila, and H. Toivonen. Interactive exploration of interesting findings in the telecommunication network alarm sequence analyzer. *Information and Software Technology*, 41(9):557–567, June 1999.

[KPTT06] V. Kapoor, P. Poncelet, F. Trousset, and M. Teisseire. Privacy preserving sequential pattern mining in distributed databases. In *Proceedings of the 15th ACM international conference on Information and knowledge management (CIKM '06)*, pages 758–767, Arlington, Virginia, USA, November 2006. ACM.

[KPWD03] H. Kum, J. Pei, W. Wang, and D. Duncan. ApproxMAP: Approximate mining of consensus sequential patterns. In D. Barbará and C. Kamath, editors, *Proceedings of the 3rd SIAM International Conference on Data Mining (SDM'03)*, pages 311–315, San Fransisco, CA, USA, May 2003. SIAM.

[KPWK08] S. Kim, S. Park, J. Won, and S. Kim. Privacy preserving data mining of sequential patterns for network traffic data. *Information Sciences: an International Journal*, 178(3):694–713, February 2008.

[LNSP07] P. Laur, R. Nock, J. Symphor, and P. Poncelet. Mining evolving data streams for frequent patterns. *Pattern Recognition*, 40(2):492–503, February 2007.

[LPFK07] L. Liao, D. J. Patterson, D.r Fox, and H. Kautz. Learning and inferring transportation routines. *Artificial Intelligence*, 171(5-6):311–331, April 2007.

[LPP98] K. J. Lang, B. A. Pearlmutter, and R. A. Price. Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In V. Honavar and G. Slutzki, editors, *Proceedings of the 4th International Colloquium on Grammatical Inference (ICGI '98)*, volume 1433 of *Lecture Notes in Computer Science*, pages 1–12, Ames, Iowa, USA, July 1998. Springer.

[McG05] K. McGarry. A survey of interestingness measures for knowledge discovery. *The Knowledge Engineering Review*, 20(3):39–61, March 2005.

[MCP98] F. Masseglia, F. Cathala, and P. Poncelet. The PSP approach for mining sequential patterns. In J. M. Zytkow and M. Quafafou, editors, *Proceedings of the 2nd European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD'98)*, volume 1510 of *Lecture Notes in Computer Science*, pages 176–184, Nantes, France, September 1998. Springer.

[MdlH04] T. Murgue and C. de la Higuera. Distances between distributions: Comparing language models. In A. Fred, T. Caelli, R.P.W. Duin, A. Campilho, and D. Ridder., editors, *Proceedings of the 10th International Workshop on Syntactical and Structural Pattern Recognition (SSPR'04)*, volume 3138 of *Lecture Notes in Computer Science*, pages 269–277, Lisbon, Portugal, August 2004. Springer.

[Mit97] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[Mob06] B. Mobasher. *Web Data Mining: Exploring Hyperlinks, Contents and Usage Data*, chapter Web Usage Mining. Springer, 2006.

[MPT03] F. Masseglia, P. Poncelet, and M. Teisseire. Incremental mining of sequential patterns in large databases. *Data and Knowledge Engineering*, 46(1):97–121, July 2003.

[MS98] N. Megiddo and R. Srikant. Discovering predictive association rules. In R. Agrawal, P. E. Stolorz, and G. Piatetsky-Shapiro, editors, *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining (KDD'98)*, pages 274–278, New York City, New York, USA, August 1998.

[MTP04] F. Masseglia, M. Teisseire, and P. Poncelet. Extraction de motifs séquentiels. problèmes et méthodes. *Ingénierie des Systèmes d'Information*, 9(3-4):183–210, 2004.

[MTV97] H. Mannila, H. Toivonen, and A. I. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289, September 1997.

[Mur06] T. Murgue. *Extraction de données et apprentissage automatique pour les sites web adaptatifs*. PhD thesis, École doctorale de Saint-Étienne, 2006.

[OG92] J. Oncina and P. García. Inferring regular languages in polynomial update time. In N. Pérez de la Blanca, A. Sanfeliu, and E. Vidal, editors, *Pattern Recognition and Image Analysis*, volume 1 of *Series in Machine Perception and Artificial Intelligence*, pages 49–61. World Scientific Publishing, 1992.

[PE97] M. Perkowitz and O. Etzioni. Adaptive web sites: an AI challenge. In Morgan Kaufmann, editor, *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*, pages 16–23, Nagoya, Aichi, Japan, August 1997.

[PH00] J. Pei and J. Han. Can we push more constraints into frequent pattern mining? In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00)*, pages 350–354, Boston, MA, USA, August 2000. ACM.

[PHP+01] J. Pei, J. Han, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings of the 17th International Conference on Data Engineering (ICDE'01)*, pages 215–224, Heidelberg, Germany, April 2001. IEEE Computer Society.

[PHW02] J. Pei, J. Han, and W. Wang. Mining sequential patterns with constraints in large databases. In *Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM'02)*, pages 18–25, McLean, Virginia, USA, November 2002. ACM.

[PZOD99] S. Parthasarathy, M. Zaki, M. Ogihara, and S. Dwarkadas. Incremental and interactive sequence mining. In *Proceedings of the 8th International Conference on Information and Knowledge Management (CIKM'99)*, pages 251–258, Kansas City, Missouri, USA, November 1999. ACM.

[Rab89] L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[RC99] J. Reagle and L. F. Cranor. The platform for privacy preferences. *Communication of the ACM*, 42(2):48–55, February 1999.

[Reb67] A. S. Reber. Implicit learning of artificial grammars. *Journal of Verbal Learning and Verbal Behavior*, 6:855–863, 1967.

[RST98] D. Ron, Y. Singer, and N. Tishby. On the learnability and usage of acyclic probabilistic finite automata. *Journal of Computer and Systeme Sciences*, 56(2):133–152, April 1998.

[SA96] R. Srikant and R. Agrawal. Mining sequential patterns: Generalizations and performance improvements. In P. M. G. Apers, M. Bouzeghoub, and

G. Gardarin, editors, *Proceedings of the 5th International Conference on Extending Database Technology (EDBT'96)*, volume 1057 of *Lecture Notes in Computer Science*, pages 3–17, Avignon, France, March 1996. Springer.

[Sch96] B. Schneier. *Applied cryptography.* John Wiley & Sons, New York, October 1996.

[Sha95] J.P. Shaffer. Multiple hypothesis testing. *Annual Review of Psychology*, 46:561–584, January 1995.

[SJ03] M. Sebban and J-C. Janodet. On state merging in grammatical inference: A statistical approach for dealing with noisy data. In T. Fawcett and N. Mishra, editors, *Proceeding of the 20th International Conference on Machine Learning (ICML'03)*, pages 688–695, Washington, DC, USA, August 2003. AAAI Press.

[SL94] D. E. Stewart and Z. Leyk. Meschach: matrix computations in C. In *Proceedings of the Centre for Mathematics and its Applications*, volume 32, 1994.

[SP01] M. Spiliopoulou and C. Pohle. Data mining for measuring and improving the success of web sites. *Data Mining and Knowledge Discovery*, 5(1/2):85–114, January 2001.

[SVC01] Y. Saygin, V. S. Verykios, and C. Clifton. Using unknowns to prevent discovery of association rules. *SIGMOD Record*, 30(4):45–54, December 2001.

[TDdlH00] F. Thollard, P. Dupont, and C. de la Higuera. Probabilistic DFA inference using kullback-leibler divergence and minimality. In P. Langley, editor, *Proceedings of the 17th International Conference on Machine Learning (ICML '00)*, pages 975–982, San Fransisco, CA, USA, July 2000. Morgan Kaufmann.

[Tho00] F. Thollard. *Inférence grammaticale probabiliste pour l'apprentissage de la syntaxe en traitement de la langue naturelle.* PhD thesis, Université Jean Monnet, 2000.

[TMF03] F. Tao, F. Murtagh, and M. Farid. Weighted association rule mining using weighted support and significance framework. In L. Getoor, T.E. Senator, P. Domingos, and C. Faloutsos, editors, *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, pages 661–666, Washington, DC, USA, August 2003. ACM.

[Val84] L. G. Valiant. A theory of the learnable. *Communication of the Association for Computing Machinery*, 27(11):1134–1142, November 1984.

[VBF+04] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis. State-of-the-art in privacy preserving data mining. *SIGMOD Record*, 33(1):50–57, March 2004.

[VC02] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of the 8th ACM SIGKDD inter-*

national conference on Knowledge Discovery and Data mining (KDD'02), pages 639–644, Edmonton, Alberta, Canada, July 2002. ACM.

[VCZ06] J. Vaidya, C. Clifton, and M. Zhu. Privacy preserving data mining. In *Advances in Information Security*, volume 19. Springer, September 2006.

[Web07] G.I. Webb. Discovering significant patterns. *Machine Learning*, 68(1):1–33, July 2007.

[Wet80] C. S. Wetherell. Probabilistic languages: A review and some open questions. *ACM Computing Surveys*, 12(4):361–379, December 1980.

[YHA03] X. Yan, J. Han, and R. Afshar. CloSpan: Mining closed sequential patterns in large datasets. In D. Barbará and C. Kamath, editors, *Proceeding of the 3rd SIAM International Conference on Data Mining (SDM'03)*, pages 166–177, San Francisco, CA, USA, May 2003. SIAM.

[YL05] U. Yun and J. J. Leggett. Wlpminer: Weighted frequent pattern mining with length-decreasing support constraints. In T. B. Ho, D. W. Cheung, and H. Liu, editors, *Proceedings of the 9th Pacific-Asia conference on advances in knowledge discovery and data mining (PAKDD'05)*, volume 3518 of *Lecture notes in computer science*, pages 555–567, Hanoi, Vietnam, May 2005. Springer.

[Yun08] U. Yun. A new framework for detecting weighted sequential patterns in large sequence databases. *Knowledge-Based Systeme*, 21(2):110–122, March 2008.

[Zak00] M. J. Zaki. Sequence mining in categorical domains: incorporating constraints. In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM'00)*, pages 422–429, McLean, VA, USA, November 2000. ACM.

[Zak01] M. J. Zaki. SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, January 2001.

[ZCM04] J. Z. Zhan, L. Chang, and S. Matwin. Privacy-preserving collaborative sequential pattern mining. In *Proceedings of SIAM International Workshop on Link Analysis, Counter-terrorism, and Privacy*, pages 61–72, Lake Buena Vista, Florida, USA, April 2004. In conjunction with SIAM International Conference on Data Mining (SDM' 04).

[ZPT04] H. Zhang, B. Padmanabhan, and A. Tuzhilin. On the discovery of significant statistical quantitative rules. In W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, editors, *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD '04)*, pages 374–383, Seattle, Washington, USA, August 2004. ACM.

[ZXML02] Q. Zheng, K. Xu, S. Ma, and W. Lv. The algorithms of updating sequential patterns. In *Proceedings of the 5th International Workshop on High Performance Data Mining (HPDM'02)*, Arlington, VA, USA, April 2002. In conjunction with the 2nd SIAM Conference on Data Mining (SDM'02).

[ZYHY07] F. Zhu, X. Yan, J. Han, and P.S. Yu. Efficient discovery of frequent approximate sequential patterns. In *Proceedings of the 7th Internatio-*

nal Conference on Data Mining (ICDM'07), pages 751–756, Omaha, Nebraska, USA, October 2007. IEEE Computer Society.

Table des figures

Liste des tableaux

Liste des Algorithmes